DARPA Adaptive Computing Systems Program[1]

Evolvable Hardware for Adaptive Computing

Final Report

March 28, 2002

Adrian Stoica, Jet Propulsion Laboratory

[1] http://ehw.jpl.nasa.gov/Documents/PDFs/ehwacs.pdf

Executive Summary

The main objective of this research was to demonstrate on-chip automated circuit reconfiguration using evolutionary techniques. A secondary objective was to develop a technology base allowing complete evolvable hardware systems to converge in seconds, orders of magnitude faster than simulation-based evolution. These objectives have been achieved. We developed and demonstrated three generations of evolution-oriented devices reconfigurable at the transistor level, implementing programmable analog, digital, and mixed-circuit solutions. On these chips we demonstrated automatic evolutionary reconfiguration in seconds, exceeding four orders of magnitude improvement in speed compared to evolutions in simulation.

Major accomplishments of this work include: a) development of three generations of evolvable chips, the latest being also the world's largest programmable analog and mixed signal array, b) development of a technology base that could be used by the research community at-large, which ranges from stand-alone/compact evolvable systems to supercomputer-based ones, c) a set of techniques that overcome difficulties in evolving systems for real-world applications, and d) identification of new application domains. The most important contributed techniques are: mixtrinsic evolution, design and reuse, and morphing through fuzzy topologies. Mixtrinsic evolution overcomes obstacles in obtaining portable and robust solutions. Design and reuse offers a solution for scalability, enabling evolution of complex circuits. Morphing through fuzzy topologies demonstrated one order of magnitude speed-up in evolution. New application domains born through this research include polymorphic electronics, which refers to circuit designs with superimposed concealed functions, and reconfiguration-based extreme electronics, which refers to reconfigurable circuit solutions to degradation of functionality due to adverse temperature effects.

While mainstream adaptive computing efforts focused on building faster and larger programmable devices, and tools that humans can use to design adaptive computing systems, our contribution is unique in providing *automatic* reconfiguration (self-adaptation) for programmable devices. With this technology demonstrated, further development steps need to be taken in order to insert it into real-world systems. This report presents several recommendations for future research and development.

Evolvable hardware technology is particularly significant to future DOD autonomous systems, providing on-board resources for reconfiguration to self-adapt to situations, environments and mission changes. It would also enable smart embeddable sensor arrays for structures such as smart skins for airplanes and submarines. In addition, this technology can be used to reduce massive amounts of sensor data to lean data sent to centralized digital systems. Evolution-derived polymorphic circuits have particular applications to security, watermarking and information exfiltration. High temperature electronics also have a wide range of far reaching DOD applications.

Table of Content

1 Introduction

1.1 Background and overview of the effort

This report summarizes the results of a three-year research project on Evolvable Hardware for Adaptive Computing. The objective of this research was to develop the technology needed to demonstrate on-chip automated circuit design/reconfiguration - thus a means of achieving adaptive computing - using evolutionary techniques. It was targeted to create the technology base that would allow compact/miniature complete evolvable hardware system to would evolve on-chip in seconds, orders of magnitude faster than simulation-based evolution.

Two main challenges of evolving electronics, in particular analog circuitry, relate to the lack of "evolution-oriented" devices (to allow fast and accurate hardware evolution instead of using simulations), and to the difficulty of translating target specifications into fitness functions for proper guidance of evolution. The first part of our effort focused on building evolution-oriented devices. Evolution requires a great number of candidate configurations to be tested, starting with randomly generated ones; randomly generated configurations may include some that could damage (burn) current programmable digital devices. On the other hand current commercial programmable analog devices lack the sufficient number of components to be interesting for evolutionary design. In the absence of such devices evolution can be attempted in simulations, yet accurate simulations (e.g. at transistor level for analog circuitry) are prohibitively slow, in particular as the complexity of the circuit increases. During this research we have designed, built and experimented with three generations of Field Programmable Transistor Arrays (FPTA), a novel type of evolution-oriented devices reconfigurable at transistor level. Over four orders of magnitude speed-up of evolution was obtained on the FPTA chip compared to SPICE simulations on a Pentium processor (this performance figure was obtained for a circuit with approximately 100 transistors; the speed-up advantage increases with the size of the circuit). The evolutionary algorithm was implemented in a DSP that directly controlled the FPTA, forming together a board-level evolvable system without overheads and bottlenecks, and on which electronic circuits evolve in seconds.

The evolutionary algorithms (EA) are used as a search method guiding the automatic synthesis/design of electronic circuits. A key performance index was to exceed four orders of magnitude speed-up of hardware evolution compared to simulations, reducing days and weeks of waiting for the result of an experiment to seconds and minutes. Thus, quasi real-time adaptation of devices can be achieved. The self-configuration achieved has special meaning since little work in "adaptive devices" was done to make them *self*-adaptive, or, if so, the embedded adaptation mechanism was very basic, with simple and *a priori* determined possible states of devices. The same was true for adaptive computing, with little been done to automate the adaptation: the states on which an adaptive computing system is determined *a priori*. In this context "self" refers to the complete system including the evolutionary algorithm, and our original plan was to integrate the evolutionary processor (EP) with the reconfigurable area on a single chip. Later however, it was decided to reduce the complexity and risk of fabrication of a complex mixed-signal chip and operate from two chips, one for EP (digital) and one for reconfigurable hardware (RH) (programmable analog/mixed signal) as long as the stand-alone system at board level preserved the advantages of high speed communication between EP and RH, albeit on a slightly less compact solution. Two bigger advantages however have surfaced: the possibility to use the EP to

control evolution of other structures, such as MEMS and reconfigurable antenna (one of which has been developed part of a concurrent effort), and allowing the RH to be controlled by other algorithms.

With the availability of a platform enabling rapid evolvable hardware experiments, the true difficulty has surfaced, i.e. translating target specifications into the language of evolution: representations, fitness function and parameters of the algorithm. One reason for the difficulty of evolving in completely autonomous systems is the necessity of providing complete up-front specifications, and completeness is not often obvious. In a computer-assisted design the human can come back and provide extra information that may have been omitted in the beginning. In an automated systems that is not possible and evolution usually finds the easiest way out.

The demonstration of evolvable hardware for synthesis of adaptive computing circuits includes example of adaptive filters, adaptive gain control, linear and nonlinear amplifiers, discriminators, etc. Such evolutions, in which tens to hundreds of thousands of circuits were evaluated and generally took only a few seconds to perform.

Several important lessons were learned during this effort. An early observation was that testing in hardware a solution evolved in simulations often led to a different response, and vice-versa, solutions determined with hardware in the loop did not simulate to same behavior. We demonstrated that we could solve this by a new method named "mixtrinsic evolution", based on a mixed population with individuals assessed both in hardware and software, where the fitness is an indication of the performance in both modes. This idea was extended to different evaluations such as at different time scales, etc. This makes a system able to cope with another trap of evolution: the fact that it often seeks the "easiest" way out, and does not "assume" what we do. For example, if fitness rewards a logical AND gate tested at nanosecond scale, the best solution found by evolution, although perfect at nanosecond scale, may not necessarily perform as AND gate at microsecond range, etc. These are some examples of difficulties in dealing with evolutionary design/adaptation, and many more open questions remain. How to partition a complex problem? What should the fitness function be? How to ensure the robustness of a solution without testing it exhaustively during evolution? What is the best way to provide specifications? What resources and how to organize them on an evolvable chip?

Our research on developing evolvable hardware technology continues with NASA funding, with a focus on enhancing hardware survivability and adaptation. The technology base developed in this effort, including the FPTA chips, code for controlling DSP running the EA, all software and hardware schematics of components involved, etc. are available to the community, and efforts toward implementation of similar systems have started at other centers, e.g. at the NASA Marshall Space Flight Center. The effort also led to several spin-off new research directions and additional funding. Evolutionary recovery of circuits exposed to extreme temperatures (over 300C) led to a new NASA-funded task, focused on expanding the operational range of devices through circuit solutions (reconfiguration). Similarly, the concept of polymorphic electronics attracted new DARPA (ATO) funding for evolutionary design of circuits with built-in superimposed functionalities.

1.2 Contents

This Final Report is organized as follows:

Section 2 *Problems and Work Objective* refers to automated design, adaptive computing and survivable hardware as open problems and specifies the objective of this work.

Section 3 *Evolution for Design and Adaptation* presents the evolutionary approach to automated design and adaptation.

Section 4 *Structure of Evolvable Systems* describes a field programmable array used as the reconfigurable hardware platform and an evolutionary algorithm that controls reconfiguration. These two components in different embodiments are combined to create various systems ranging from a stand-alone system with the algorithm running on a DSP to larger testbeds that include a supercomputer.

Section 5. *Illustrative Experiments* focuses on selected experiments to illustrate some key characteristics of evolution in hardware and software.

Section 6 *Achievements and Significance* overviews novel techniques developed during this effort, new directions opened by this research and some of the lessons learned.

Section 7 *Vision and Recommendations for Future Research* offers a vision of evolvable adaptive infrastructures of tomorrow, comments on some of the main challenges and open questions and proposes new directions of research.

Section 8 *Concluding Remarks* offers the perspective that evolutionary engineering will mean a shift from engineering design to engineering specification (fitness) design.

The *References* section gives details of the publications during the period of this contract.

The Appendices offer additional information on

- A. Objective and Statement of Work: Proposed and Accomplished (There, it is explained how each item has been accomplished, as promised.)
- B. Field Programmable Transistor Array (FPTA) chip
- C. Evolutionary Processor
- D. Stand-alone board-level evolvable system
- E. Supercomputer code
- F. Documentation for a series of evolved circuits
- G. Other EHW Testbeds

Two CDs supplement this report, the first contains all related publications, code, and other documentation such as chip and board schematics, the second contains video demonstrations of several experiments described in this report. The complete information on the chips (including layout, etc) is available on our website at [EHW_WEB].

2 Problems and Work Objective:

Automatic Programming (AP), in the sense of automatic software generation from a set of specifications, can be seen as the Holy Grail of computer research. AP would be a great empowerment to humans, who no longer would need to learn specialized languages to control the increasingly more computerized world around them, but will only have to communicate higher-level requirements goals expressed in natural language. On a different perspective, autonomous systems need a way to change their behaviors and adapt to new situations without human interventions. This requirement includes systems operating in space or in the ocean, silent (no communication) missions, or scout robotic systems that have no possibility of interactions with humans. **Automated design/reconfiguration** of a programmable device is very similar to automatic programming.

Adaptive computing (AC) systems aim at morphing themselves to best adapt to the problem to be solved. In a more general sense AC includes adaptation to environments and constraints such as power. Field Programmable devices provide support for adaptive computing. The focus of this field has been on building faster and larger devices, and on tools that would empower human designers to more easily and rapidly create designs. Mechanisms to provide automatic adaptation are lacking, although this aspect is the very essence of the concept.

A variety of space and military applications require **survivable and flexible hardware** that would enable long durations missions and operations in harsh environments. Future space missions use concepts of spacecraft surviving in excess of 100 years. Temperatures over 460°C as encountered on the surface of Venus pose challenges to current electronics. *Survivability* means the hardware is able to cope with changes due to temperature, radiation, aging or malfunctions. *Flexibility* means that the hardware could adapt to mission and environmental changes, as well as optimal changes to save resources, such as energy savings, and maximize utility, such as optimal information processing. All this requires a new form of self-configurable hardware.

Analog is what we get from sensors around us, and the more sensors we use and the more performance we desire, the more bottlenecks we will have in digital unless we do more analog computing to provide leaner data. In principle efficient problem solving, faster and in less power may be performed in analog if we could address programmability, precision and drift. Smart skins and bio-interfaces are examples of areas that would largely benefit from compact, low-power **analog computing structures**.

The research described in the following addresses all the above areas. Evolvable Hardware is a technology for evolutionary automatic design and adaptation/reconfiguration/reprogramming of programmable devices, which enables adaptive computing and offers enhanced flexibility and survivability. The development of evolution-oriented analog devices was a special focus of this effort. The devices enable evolution in hardware with considerable improvements in the time for evolution, accuracy of results and scalability as compared to evolution in software.

3 Approach: Evolution for Design and Adaptation

The idea behind evolutionary circuit synthesis/design and evolvable hardware (EHW) is to employ a genetic search/optimization algorithm that operates in the space of all possible circuits and determines solution circuits with desired functional response (here the word synthesis is used in most general sense). The genetic search is tightly coupled with a coded representation of the candidate circuits. Each circuit gets associated a "genetic code" or chromosome; the simplest representation of a chromosome is a binary string, a succession of 0s and 1s that encode a circuit. Synthesis is the search in the chromosome space for the solution corresponding to a circuit with a desired functional response. The genetic search follows a "generate and test" strategy: a population of candidate solutions is maintained each time; the corresponding circuits are then evaluated and the best candidates are selected and reproduced in a subsequent generation, until a performance goal is reached. Circuit evaluation can be done on software models of the hardware using circuit simulators, in which case evolution is called *extrinsic* evolution, or directly in reconfigurable hardware, in which case it is called *intrinsic*. The main steps of evolutionary synthesis are illustrated in Figure 1.

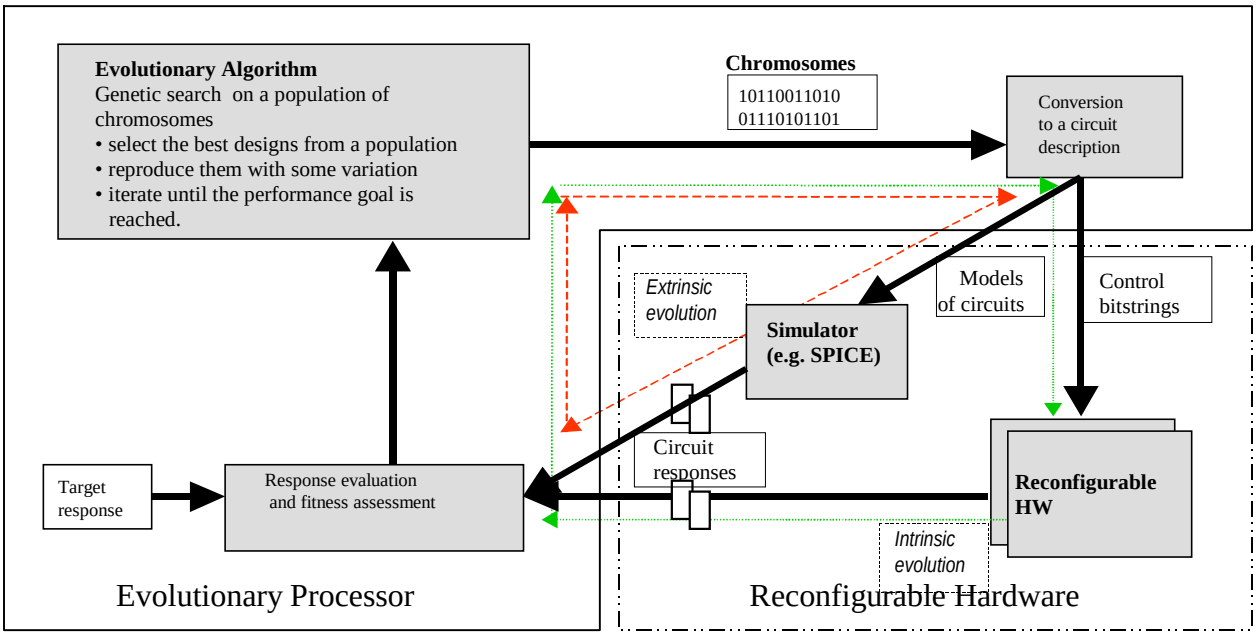


Figure 1 Main steps in evolutionary synthesis.

At first, a population of chromosomes is randomly generated. The chromosomes are converted into circuit models (extrinsic EHW) or control bitstrings downloaded to programmable hardware (intrinsic EHW). Circuit responses are compared against specifications of a target response, and individuals are ranked based on how close they come to satisfying it. Preparing for a new iteration loop, a new population of individuals is generated from the pool of best individuals in the previous generation. This is subject to a generally random selection of individuals from a best

individuals pool, random swapping of parts of their chromosomes (crossover operation) and random flipping of chromosome bits (mutation operation). The process is repeated for several generations, resulting in increasingly better individuals. The randomness helps to avoid being trapped in local optima. Monotonic convergence (in a loose Pareto sense) can be forced by unaltered transference to the next generation of the best individual from the previous generation. There is no theoretical guarantee that the global optimum will be reached in a useful amount of time; however evolutionary/genetic search is considered by many to be the best choice for very large, highly unknown search spaces. The search process is usually stopped after a number of generations, or when closeness to the target response has reached a sufficient degree. One or several solutions may be found among the individuals of the last generation

Choices such as representations, population size and algorithmic parameters have influence on success of convergence, yet there are no good rules on how to choose them. The optimal set of on-chip resources, configuration granularity and where to collect outputs are also open questions. Finally, converting specifications to a fitness function, currently a human controlled step, can easily allow the introduction of errors and distortion of meaning.

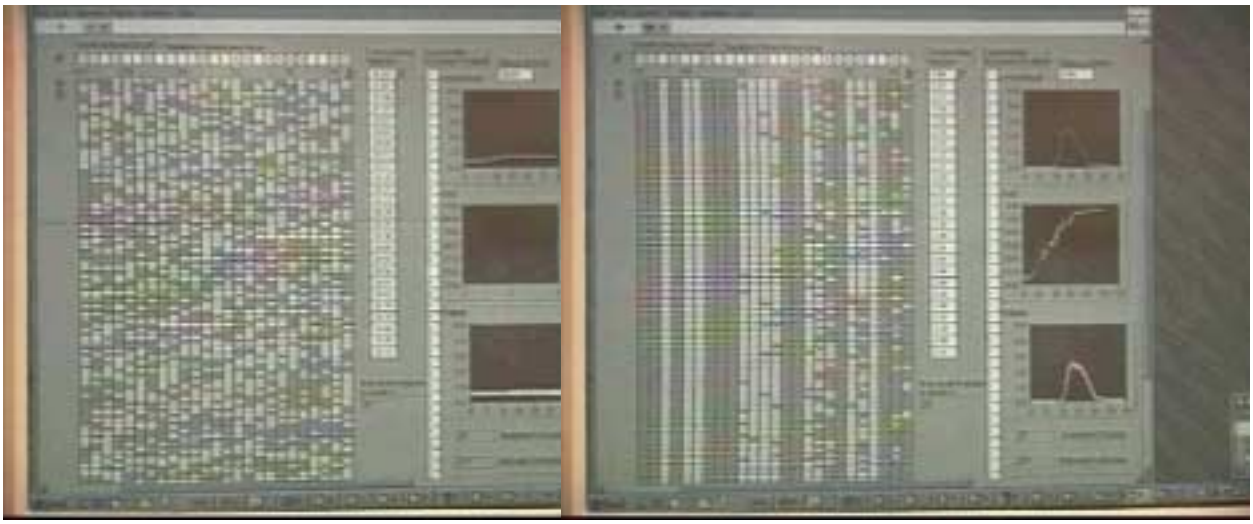


Figure 2 An example of a genome at initialization (left) finally converging to a set of mutants near a satisfactory solution (right). Each row is a chromosome for one candidate circuit (FPTA-0, 24 switches). White (black) dots indicate open (closed) switches.

4 Structure of Evolvable Systems

An evolvable hardware system is constituted of two main components: reconfigurable hardware (RH) and an evolutionary processor (EP) that acts as a reconfiguration mechanism. In several evolvable systems we built for this effort the EP was implemented and ran on a stand-alone DSP board, on a PC CPU or as a parallel implementation on a 128-node supercomputer. The RH was embodied in the form of a field programmable transistor (FPTA) array architecture, which was tested as three generations FPTA-0, FPTA-1 and FPTA-2, both in silicon as custom made chips, and as SPICE models, as well as unconstrain architectures tested in SPICE. Each combination of an EP and RH embodiment made a complete evolvable system. This section will refer to the general characteristics of the two components and will briefly describe two of the evolvable systems. More details are available in the Appendix C and D.

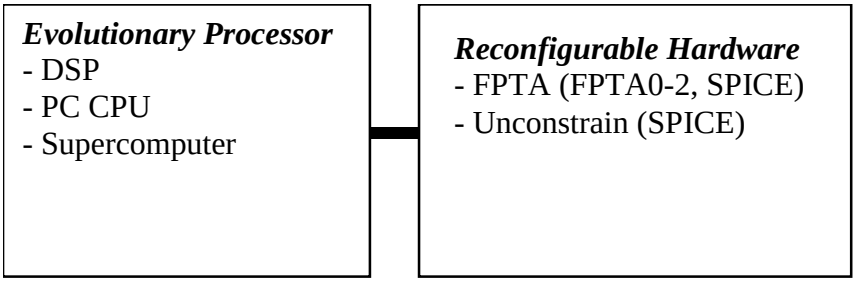


Figure 3 A block diagram of an EHW system.

4.1 The FPTA

The FPTA is an evolution-oriented reconfigurable architecture (EORA) [Stoica_IEEEJour01]. Important characteristics needed by evolution-oriented devices are *total accessibility*, needed in order to provide evolutionary algorithms the flexibility of testing in hardware any chromosomal arrangements, some of which may be dangerous for existing commercial devices (may lead to internal bus allocation conflicts and burn the chip) and thus forbidden, *granularity at low level* (here transistor) allowing evolution to choose/construct the most suitable building block for certain system, and *transparency,* which enables users to have access to internal device information, etc.

The Field Programmable Transistor Array (FPTA) is such an EORA with configurable granularity at the transistor level. It can map analog, digital and mixed signal circuits. The architecture is cellular, with each cell having a set of transistors, which can be interconnected by other "configuration transistors". For brevity, the "configuration transistors" are called switches. However, unlike conventional switches, these can be controlled for partial opening, with appropriate voltage control on the gates, thus allowing for transistor-resistor type topologies.

Cells are interconnected to local neighbors with switches. The original cell, shown in Figure 4, was implemented in the first FPTA chip (FPTA-0) and had 8 transistors interconnected by 24 switches. A variety of simple circuits can be mapped onto this device or on multiple devices by cascading them. Its design was inspired by observing a variety of analog designs in which transistors often come in rows of pairs of transistors (for various current mirrors, differential pairs etc.), and have an average of four rows between VDD and ground. More rows can be ensured cascading cells, while fewer rows can be mapped by closing some switches to bypass rows.

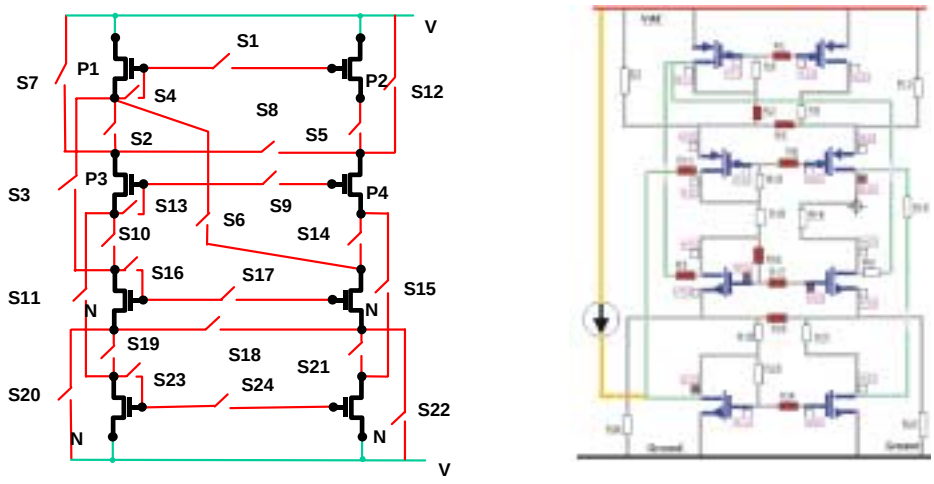


Figure 4 a) FPTA-0 cell schematic and b) LabView representation – open (closed) switches are shown as empty (filled) boxes.

The latest chip, FPTA-2, consists of an 8x8 array of re-configurable cells. Each cell has a transistor array as well as a set of programmable resources, including programmable resistors and static capacitors. Figure 5 provides a broad view of the chip architecture together with a detailed view of the reconfigurable transistor array cell. The re-configurable circuitry consists of 14 transistors connected through 44 switches. The re-configurable circuitry is able to implement different building blocks for analog processing, such as two and three stages OpAmps, logarithmic photo detectors, or Gaussian computational circuits. It includes three capacitors, Cm1, Cm2 and Cc, of 100fF, 100fF and 5pF respectively. Details of the FPTA can be found in Appendix B. and in [Stoica_EH01a], [Stoica_IEEEJour01]. Control signals come on the 9-bit address bus and 16-bit data bus, and access each individual cell providing the addressing mechanism for downloading the bit-string configuration of each cell. A total of ~5000 bits is used to program the whole chip. The pattern of interconnection between cells is similar to the one used in commercial FPGAs: each cell interconnects with its north, south, east and west neighbors. This is the first mixed-signal programmable array, FPMA, in the sense that its cells can be configured as both analog and digital circuitry; with its 64 cells it can configure more Operational Amplifiers (OpAmps) than the largest commercial Field Programmable Analog Array (FPAA) chip (which contains only 20 OpAmps).

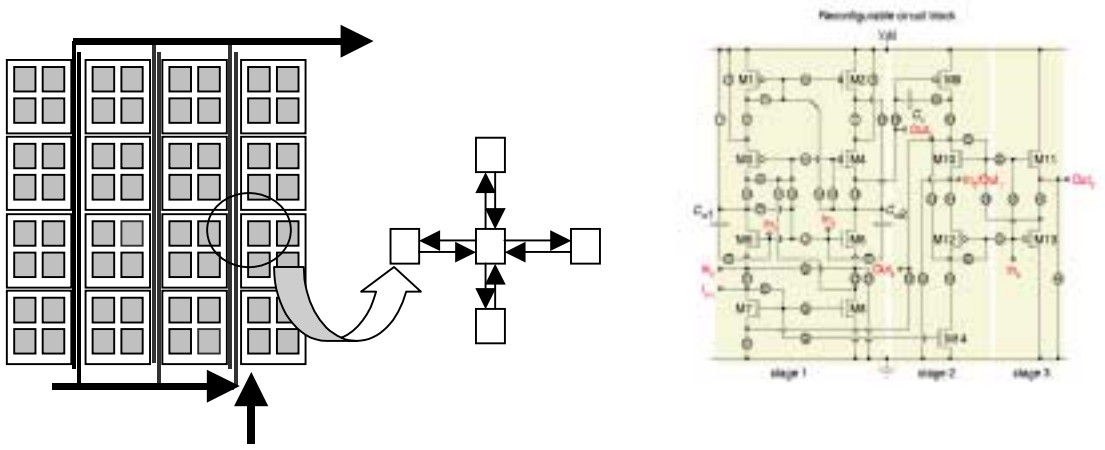


Figure 5 FPTA 2 architecture (left) and schematic of cell transistor array (right). The cell contains additional capacitors and programmable resistors (not shown), see Appendix B.

4.2 The EP

An evolutionary processor (EP) is a collection of architecture independent routines to perform tasks related to evolution in hardware. These include the baseline genetic algorithm functions (sort, crossover, mutation) as well as the I/O functions related to stimulation and evaluation of individuals.

A simple block diagram of operations taking place in a genetic algorithm (GA) is illustrated in Figure 6.

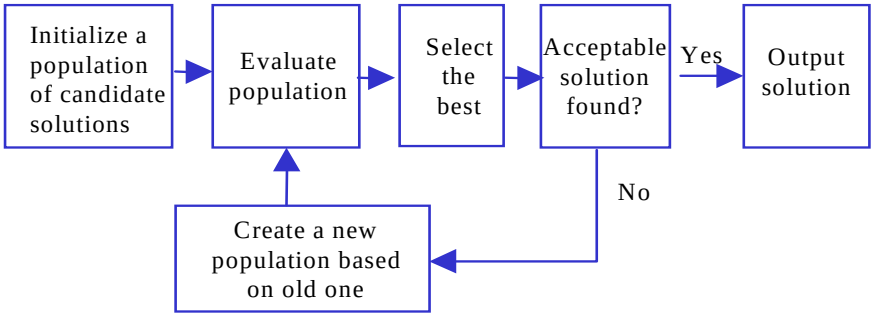


Figure 6 Block diagram of a simple Genetic Algorithm.

The time bottleneck in evolvable systems is usually determined by the evaluation of candidate solutions. However, for short evaluation times such as those needed for evolving certain electronics, fast implementations of operators used by evolutionary algorithms are needed. A detailed implementation of an EP is given in Appendix C. The rationale of a chip dedicated to the genetic search engine is to provide the required speedup, but also to avoid communication bottlenecks between the search engine and reconfigurable circuitry.

4.3 A stand-alone board-level evolvable system

A complete stand-alone board-level evolvable system was built by integrating the FPTA2 and a DSP implementing the EP as shown in Figure 7. The system is connected to the PC only for the purpose of receiving specifications and communicating back the result of evolution for analysis. The system fits in a box 8" x 8" x 3". Communication between DSP and FPTA is very fast with a 32-bit bus operating at 7.5MHz. The evaluation time depends on the tests performed on the circuit. Many of the tests attempted here require less than two milliseconds per individual, and runs of populations of 100 to 200 generations require only 20 seconds. The bottleneck is now related to the complexity of the circuit and its intrinsic response time.

Figure 7 Block diagram of a simple stand-alone evolvable system.

4.4 Testbenches

The stand-alone system is only one of the evolvable systems we use for experiments in HW evolution. A more complete testbench integrates hardware resources and a supercomputer allowing evolutions to either takes place in simulation (extrinsic), directly by on-chip reconfigurations (intrinsic) or in mixed mode, as in mixtrinsic evolution. Mixtrinsic evolution describes a situation where within the same population some individuals are evaluated in software and some in hardware. A diagram of this testbed is shown in Figure 8.

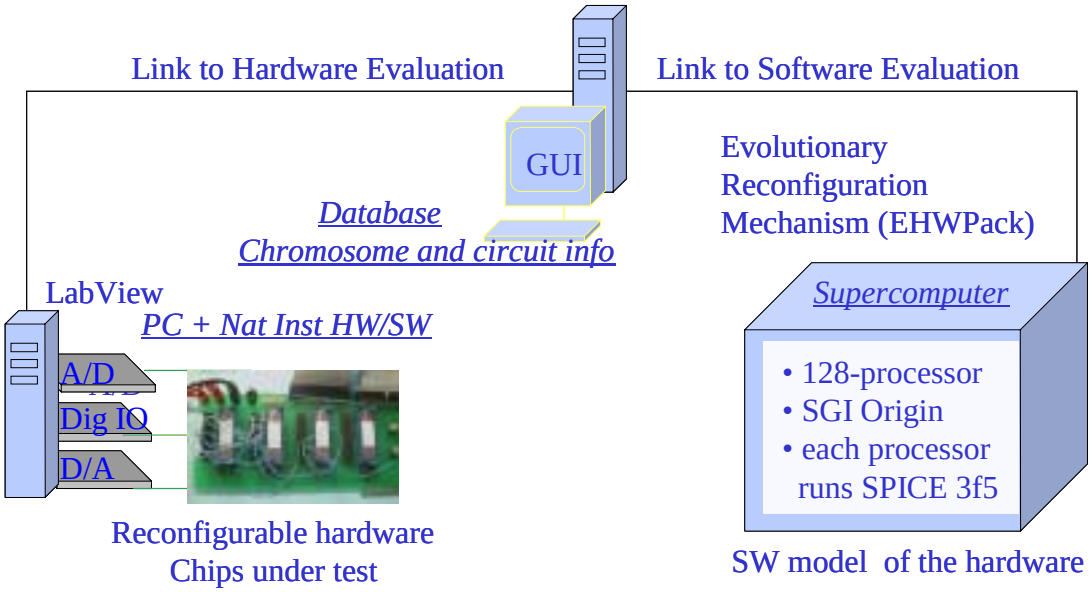


Figure 8 A more complex evolvable system/testbed

The effects of temperature were studied using a temperature testbed. At the lower temperature scale we inserted the probe in liquid nitrogen at –196°C. The heating system used produces a stream of air at temperatures up to 1000°C. Our high temperature tests focused on temperatures between 250°C and 350°C. The temperature testbed is detailed in appendix G.

5 Illustrative Experiments

A variety of circuits were evolved both in simulations and directly in hardware. Examples include Analog Computational Circuits: (Fuzzy logic circuits, Multiplier, Rectifier), Filters, Digital circuits, D/A Converters, Adaptive circuits, Polymorphic circuits, Adaptive Gain Control. Some of these evolutions are presented in Appendix F.

5.1 A detailed example: evolution of a half-wave rectifier

As described above, the EHW model implemented on the stand-alone system provides a platform for very fast evaluation of candidate solutions. This, in turn, enables us to perform evolution orders of magnitude faster than in the past. As an example of this evolution, we describe here in some detail the methodology and rationale behind one particular experiment.

The objective of this experiment is to evolve a halfwave rectifier for a 2kHz sine wave of amplitude 2V using a small portion of the FPTA. A fitness function was selected that rewards those individuals exhibiting behavior similar to a rectifier and penalize those, which do not, a simple sum of differences between the response of the circuit $R(t_s)$ and the stimulus $S(t_s)$. After evaluation of all individuals, they are sorted according to fitness and a portion (elite percentage) is set aside, $\mathcal{E}$, the remaining individuals, $\mathcal{R}$, undergo crossover with a selected individual either from the elite portion or those in $\mathcal{R}$. Mutation is then performed on the $\mathcal{R}$ individuals and the new population is formed as the union of $\mathcal{E}$ and $\mathcal{R}$. In this way the generations progress until a solution is found.

The evolutionary cycle is depicted in Figure 9 and a demonstration of how an evolution progresses is shown in Figure 10. The first panel shows the best individual of the initial population. Panels b - d show the best after 5, 20 and 40 generations, respectively.

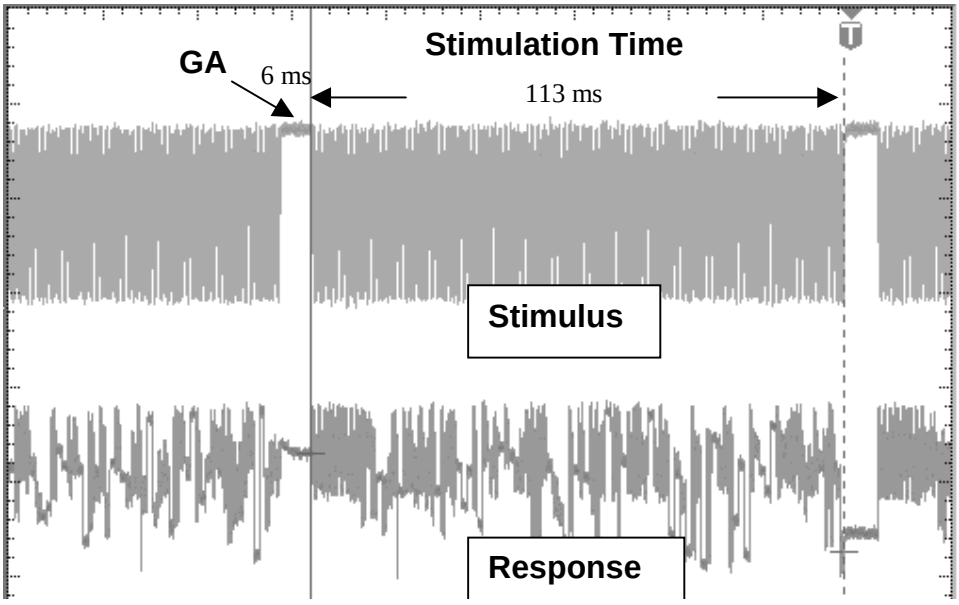


Figure 9 An example of a full GA cycle including stimulus/response (113ms) and the generation of the next generation (6ms).

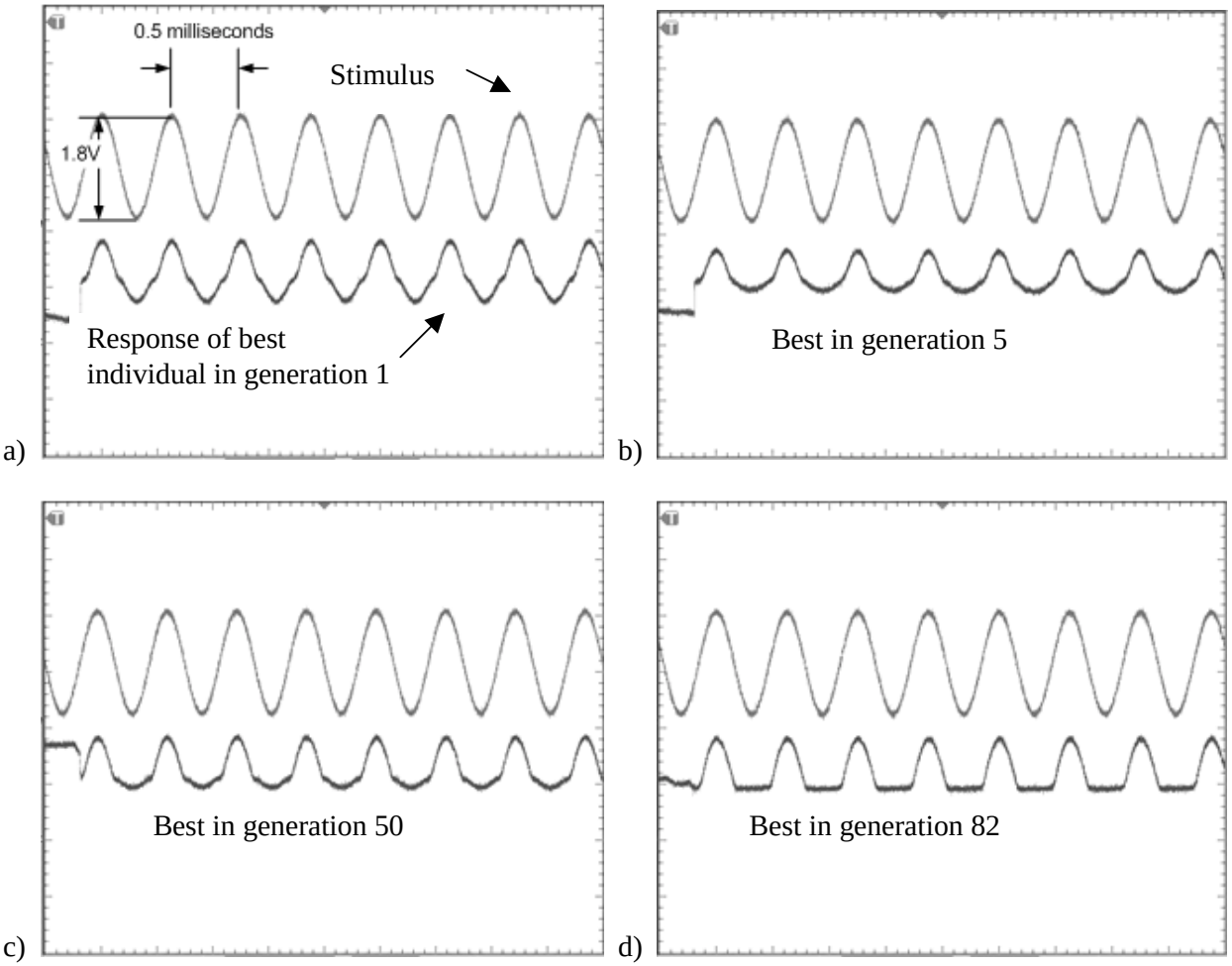


Figure 10 Evolution of a halfwave rectifier showing the response of the best individual of generation a) 1, b) 5, c) 50 and finally the solution at generation d) 82.

The final configuration can be seen represented in Figure 11 as a LabView graphical representation of the schematic in Figure 5 (right). The open (closed) switches are shown as empty (filled) boxes.

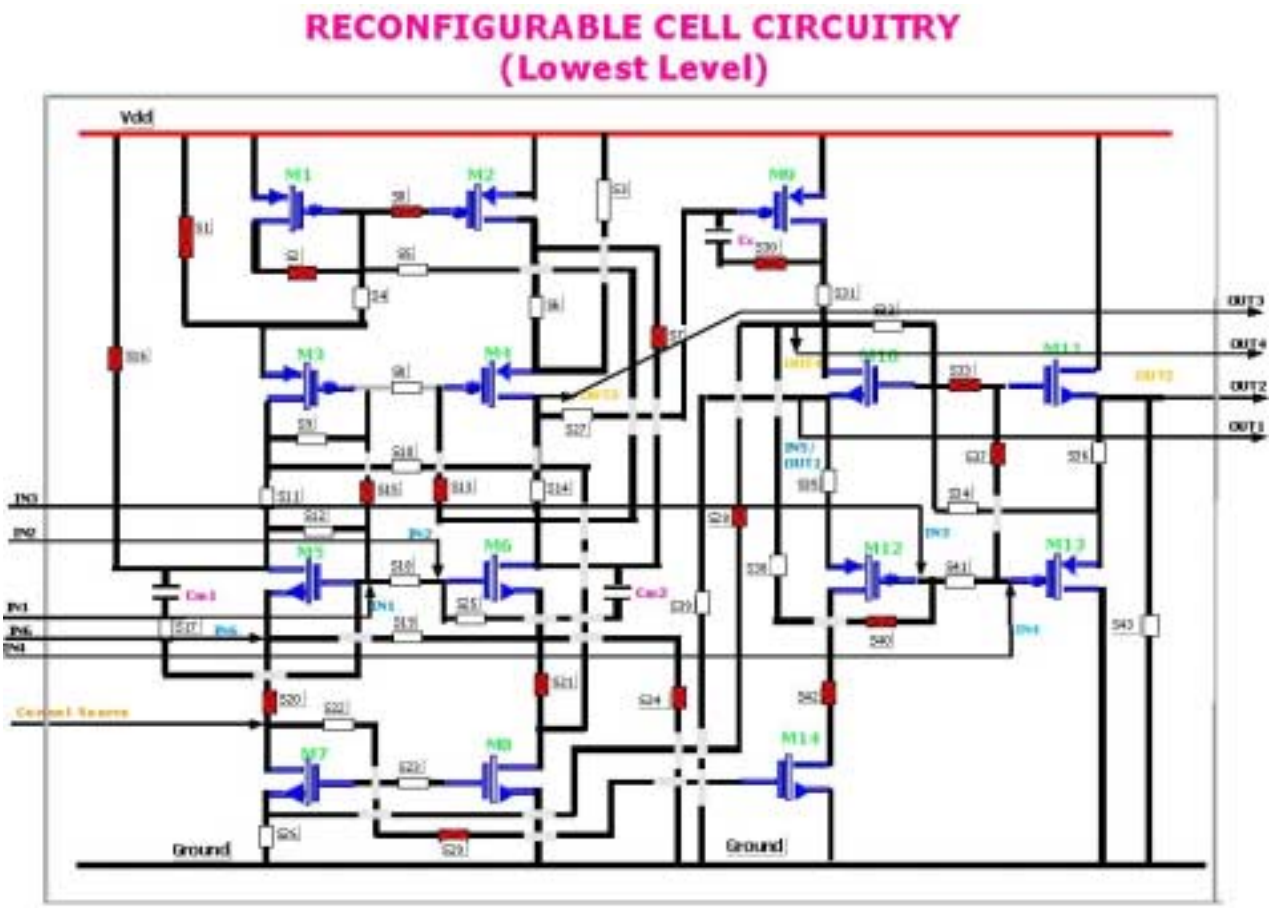


Figure 11 Diagram of an FPTA cell reconfigurable transistor array with the final configuration for the halfwave rectifier indicated by filled (closed) found after 82 generations. This diagram corresponds to the circuit in Figure 5 (right).

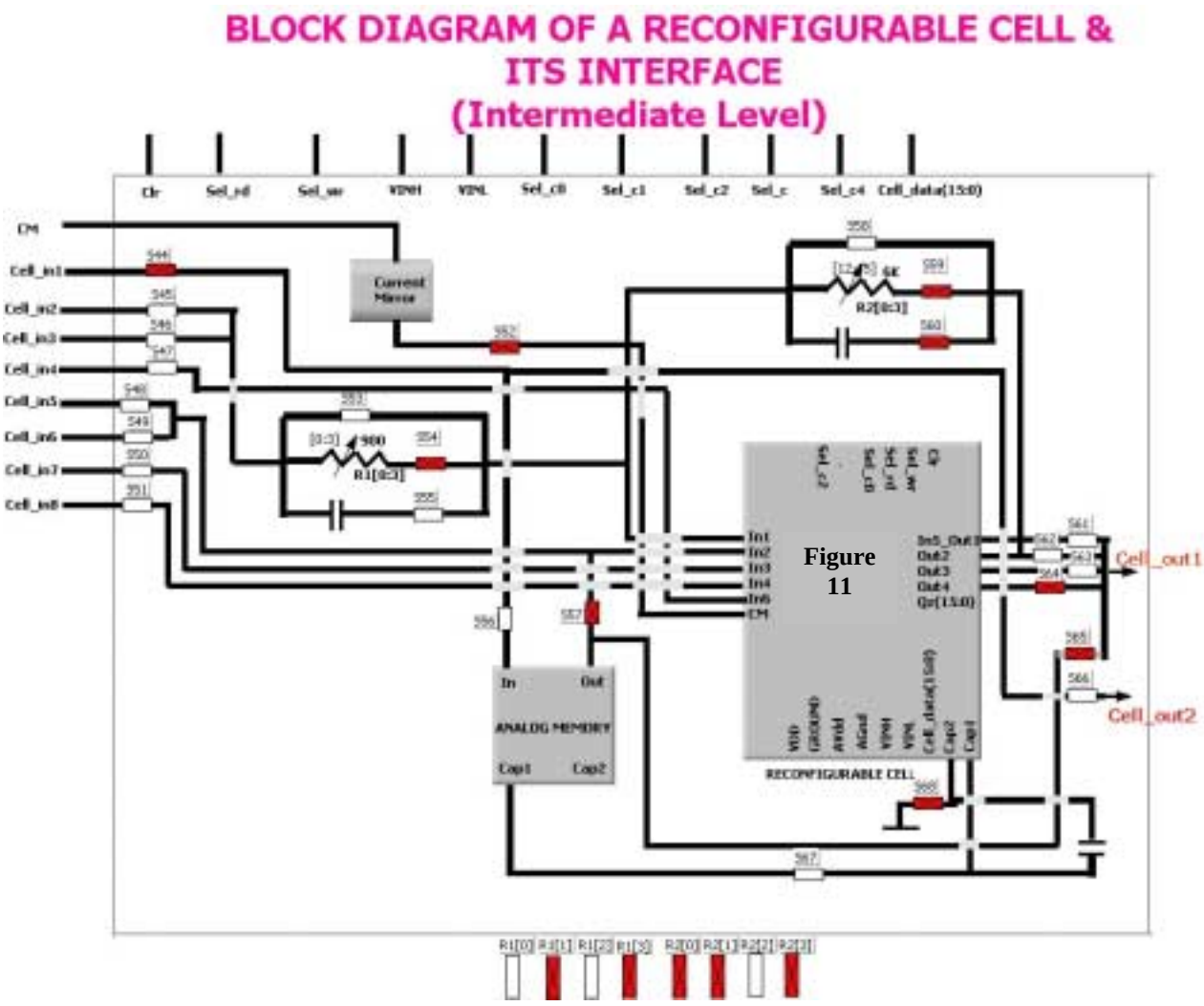


Figure 12 The cell hierarchy for the evolved halfwave rectifier.

5.2 Important Observations

5.2.1 Transients, stability and memory effects

While the halfwave rectifier objective was met for all runs, there is also confirmation of two earlier observations, illustrates artifacts of dealing with hardware. These refer to transient behaviors and FPTA-state dependence, both illustrated in Figure 13. The transient behavior describes an effect of configuration that is not stable as a function of time, whereas FPTA state dependence describes a configuration whose behavior depends on the previous configuration(s). We see both of these behaviors in the figure; most obviously we see that the function, which starts out looking similar to a halfwave rectifier ends up looking quite different. The transient in this case occurred on a time-scale of about 1 second. The second behavior must be inferred;

since this individual was selected as a solution, this means that during the initial evaluation over a 2 millisecond period, it matched quite well with the expected function, but even in part a) of the figure this does not behave sufficiently like the target rectifier, so the behavior exhibited in the initial evaluation must have been influenced by the previously downloaded configuration(s). This observed behavior is explained by the fact that there are parasitic as well as static capacitors in the chip, which can be charged during one configuration period and not discharged before the next configuration is tested, leading to an undefined charge on capacitors and subsequently altering the behavior of the circuit. In practice, this problem can be resolved by reevaluating the individuals for a longer time period before declaring success.

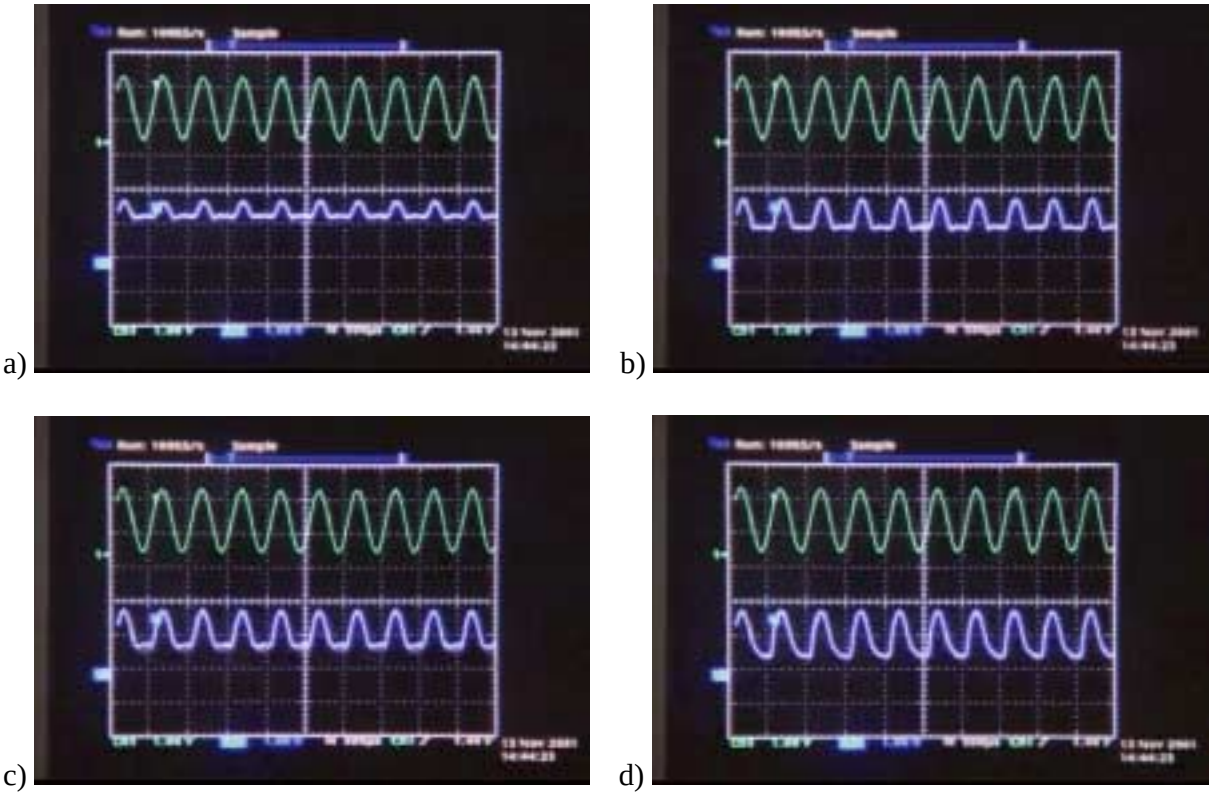


Figure 13 An example of transient behavior. The degradation shown from a) to d) occurred over the span of approximately 1 second.

5.2.2 Time constants

The importance of timescale used for circuit evaluation is critical for the achievement as observed, for example, during the experiments for the evolution of logic gates. In one particular experiment (evolution of NAND gates), the SPICE transient analysis was used to assess the circuit response. The transient analysis timescale was in the range of microseconds. The correct behavior for this timescale was quickly achieved by evolution. However, an incorrect behavior was observed when the same circuit was simulated after evolution in the timescale of seconds. This is illustrated in Figure 14, where the responses at the two timescales are depicted for one of the deceptive circuits. It has been observed that when the timescale is increased, the response for

one particular set of inputs (‘1’ and ‘0’) goes to the wrong value. In the timescale of microseconds a transient response was evaluated, and the circuit performed correctly. However, the steady state response of the circuit is actually wrong as shown in the figure in the right column.

Conversely, if we attempt to solve this problem by evaluating each circuit using a large timescale, we often obtained slow gates, since the evolutionary algorithm did not test their behavior on small timescales.

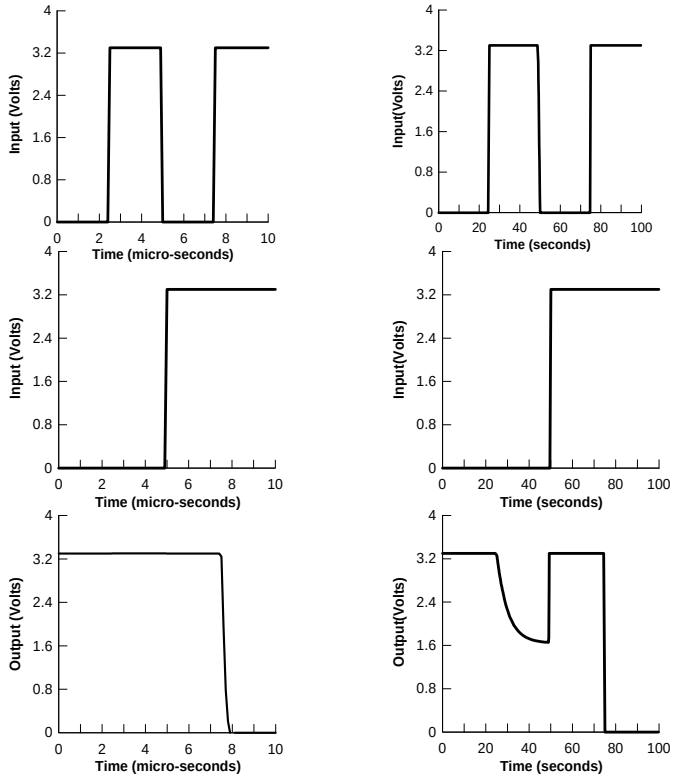


Figure 14 Evolved NAND gate evaluated in the timescale of microseconds (until 10^{-5} sec) shown in the left. Incorrect behavior of the gate when it is simulated in the timescale of seconds (until 100 seconds) is shown in the right. A capacitive load of 10pF was used at the gate output.

Performing a two-transient analysis for each candidate circuit during evolution, the first on a small timescale and the second on a larger timescale has solved this problem. For each circuit, the combined fitness measure was chosen to be the worse between the two evaluations, so that the genetic algorithm was driven to achieve a correct behavior at both timescales. Figure 15 depicts the response of circuit evolved using this method and shows a correct response at both two timescales.

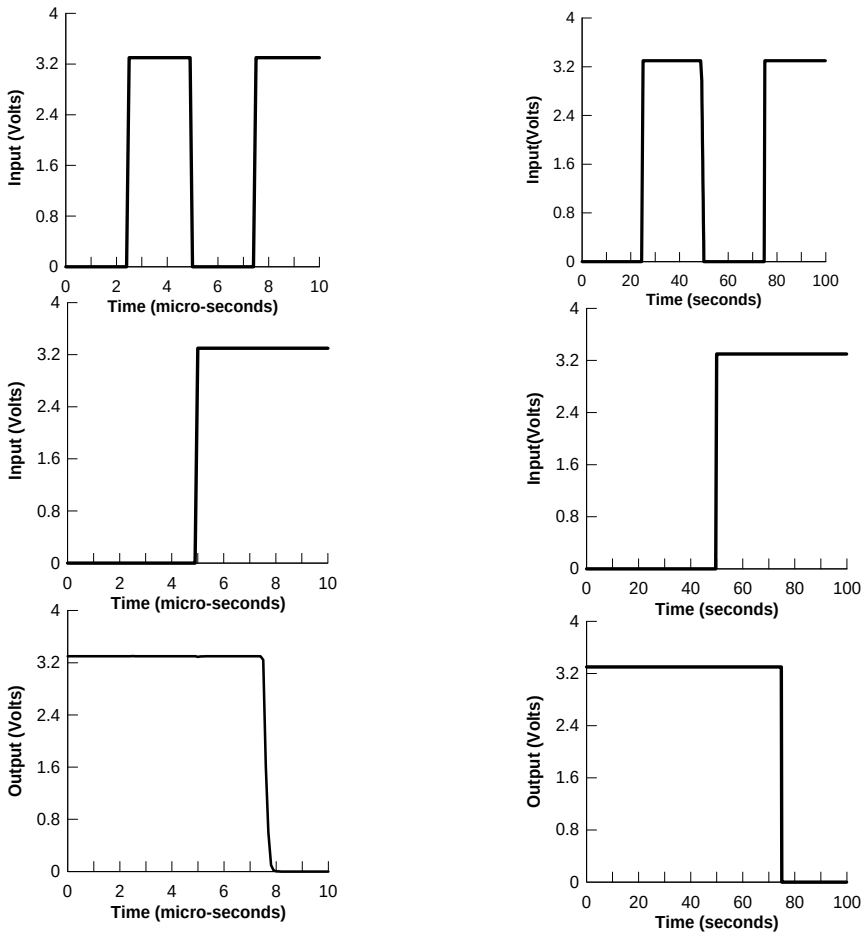


Figure 15 Evolved NAND gate using two different timescales (micro-seconds in the left and seconds in the right) show a correct behavior.

5.2.3 Mutant Solutions increased robustness of fault-tolerant system

An observation with important consequences for increased system fault tolerance is related to the behavior of mutants, individuals in the population that differ only in a few bits from the best solution in that generation. For example, in the evolution of a Gaussian circuit using FPTA-0 with 24 switches, the search led to families of circuits that differed only by a few bits in there genetic code. This is illustrated in Figure 2 (right) where vertical lines of the same color indicate switches that are the same in all best performing circuits.

The response of four mutants obtained by evolution after 50 generations with a population of 100 individuals is illustrated in the screen capture shown in Figure 16 (LabView display of the signals captured by the data acquisition boards). Notice the “mutations” in the genetic code of the solutions obtained by evolution (vertical chromosomes R24 to R1 reading from top to bottom, corresponding to switches S24 to S1 in Figure 4) compared with the human-designed circuit (rightmost vertical string).

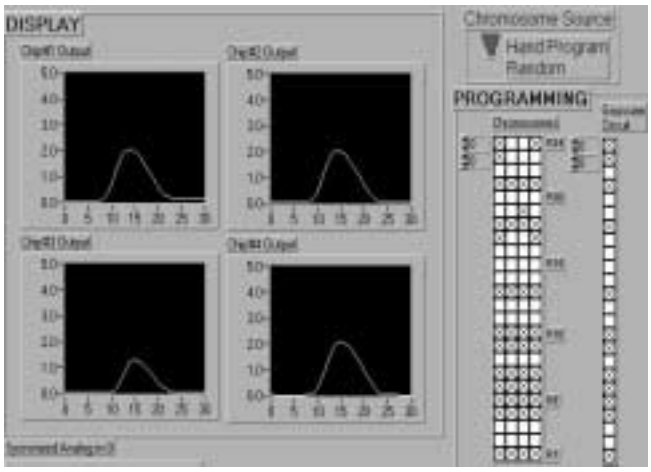


Figure 16 The "Gaussian" response of four "mutants" and their "genetic code" compared to the code of a human-designed circuit.

As indicated, the mutants play an important role in enhancing a circuit's fault-tolerance. For example, if we simulate damage to the solution circuit (the one with the best fitness in the population) by removing some wires interconnecting 2 cells of FPTA-0 that compose the solution, the response deteriorates. However, if the last generation of the evolutionary process that led to the damaged solution is still available, mutants could provide solutions, which although not optimal, would at least in part satisfy the requirements. The mutants provide a graceful degradation as, shown in Figure 17, by only a 20-30% reduction in the fitness value. A faster recovery takes place since the search for a new solution starts from the neighborhood of the old solution.

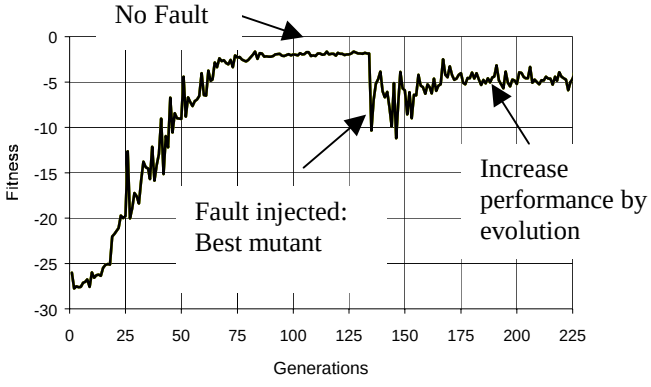


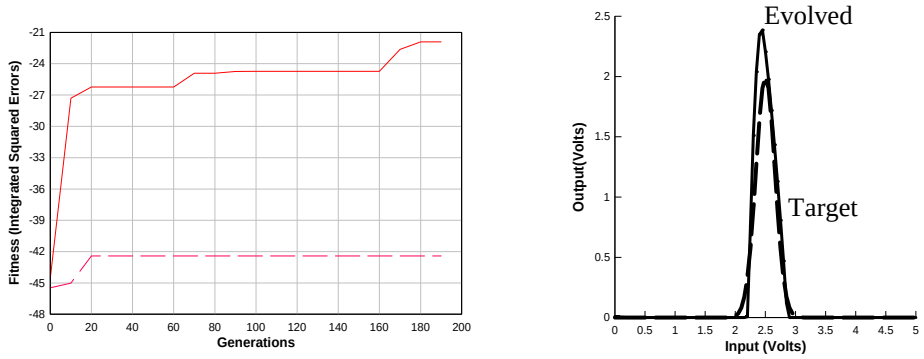
Figure 17 Fitness value monitoring the performance of the circuit. At generation 130, a fault was injected by removing an external wire between connected, single-cell FPTA chips.

This phenomenon increases the robustness of GA's when applied to this type of search, compared to a one-candidate-solution-at-a-time search method, such as simple hill climbing or simulated annealing [Keymeulen_IEEEJour00]. This is due to the fact that GA's work by accumulating fitness statistics over many generations. Moreover, the GA's allow the use of fitness statistics over many generations to create a population with mutants insensitive to faults, as opposed to designing explicitly the fault-tolerant requirement into the fitness function. The mutants perform the task adequately in the presence of a fault without the need of previous knowledge of the faults that may occur in the circuit during its lifetime.

5.2.4 Single- vs. Multiple-input Excitation

Some experiments show that excitation using multiple-inputs may lead to better results than single-input (arbitrarily defined) excitation. The following example shows the evolution of a computational circuit with a Gaussian output. This circuit uses a ramp signal as input. Two sets of experiments were performed: the first one using four circuit points at which the ramp signal was applied; and the second one in which the ramp was applied to a single point in the FPTA cell. Figure 18(a) compares the performance of the two experiments, plotting the average fitness of the best individual over 4 Genetic Algorithm (GA) executions. Each GA execution sampled 128 individuals along 200 generations. The comparison made in Figure 18(a) clearly shows that the GA performance is greatly improved if we allow more than one excitatory input point.

Figure 18(b) shows the response of the best individual achieved using multiple inputs.



a) b)

Figure 18 a) Comparison between the fitness along the generations for one-input (traces) and multiple inputs (full line) experiment for a Gaussian circuit (left graph). The fitness is the integrated squared error to the target response. B) The circuit response for multiple inputs (full line) is compared to the target (traces) in the graph at the right.

6 Achievements and Significance

A set of techniques developed during this effort, which overcome some of the known challenges of evolving systems as well as some new ones, are overviewed in the first part of this section. The second part refers to application domains opened by this research. In particular it describes polymorphic electronics, which is a new application area related to circuits with multiple superimposed functionalities and which evolution appears to be the only means of design. Another application area described here is reconfiguration-based temperature tolerant electronics.

6.1 Novel techniques

6.1.1 Mixtrinsic Evolution

The solutions obtained by evolutionary design may be plagued by what is referred to in the following as the *portability problem*. Solutions evolved in SW may behave differently when ported to HW and vice-versa. This mismatch is mainly related to simulator limitations, including the accuracy of the circuit model, incomplete information of the fabrication process, convergence problems, initial circuit conditions, etc. Failure to port solutions evolved in HW to software may also be related to the failure to model, in simulation, certain conditions of the hardware experiments such as output loads and the initial charge of parasitic capacitances.

To solve this portability problem, a new approach to EHW was developed called Mixtrinsic Evolution (ME), presented in detail in [Stoica_ICES00]. Mixtrinsic EHW encompasses a family of techniques for a variety of ways of combining intrinsic (HW) and extrinsic (SW) modes[2]. The most straightforward alternative is to evaluate each individual both in hardware and in software and assign it a combined fitness function, e.g. an average of the individual fitness for hardware and software evaluation. This constrains evolution to a solution that jointly simulates well in SW, and performs well in HW. The portability problem and the Mixtrinsic Evolution are illustrated here in an experiment where an AND gate is evolved. Figure 19 presents a circuit evolved in HW with an invalid response in SW. Figure 20 depicts a circuit evolved in SW with an invalid response in HW. Finally Figure 21 shows a circuit achieved through ME that behaves correctly both in HW and in SW.

[2] More generally, mixtrinsic refers to using a combined fitness measure derived from testing the solution in two or more different conditions, software, hardware, low/high temperatures, fast/slow time constants.

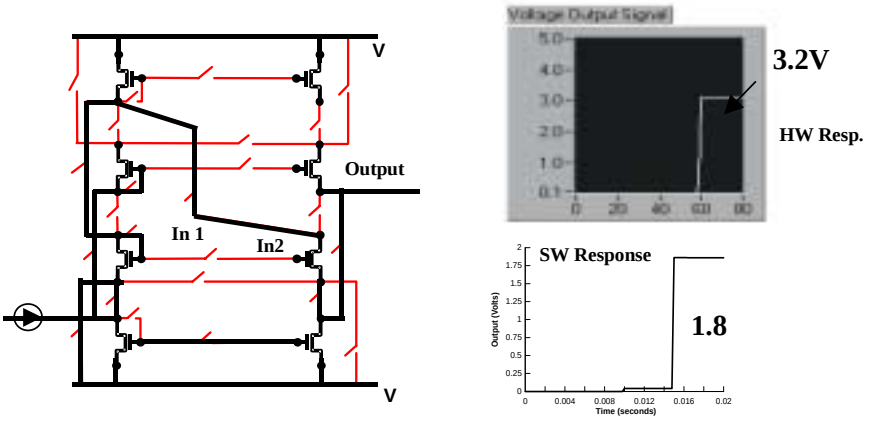


Figure 19 Intrinsically evolved circuit, its response in HW and its invalid response in SW

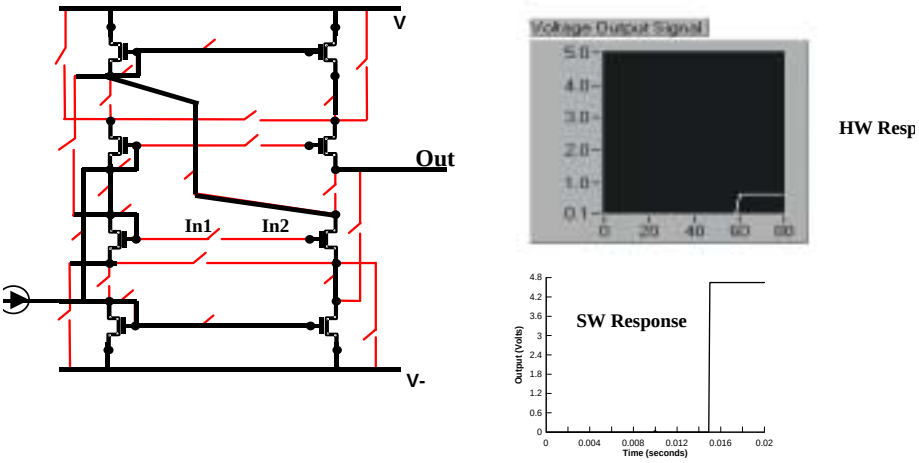


Figure 20 - Extrinsically evolved circuit, its response in SW and invalid response in HW

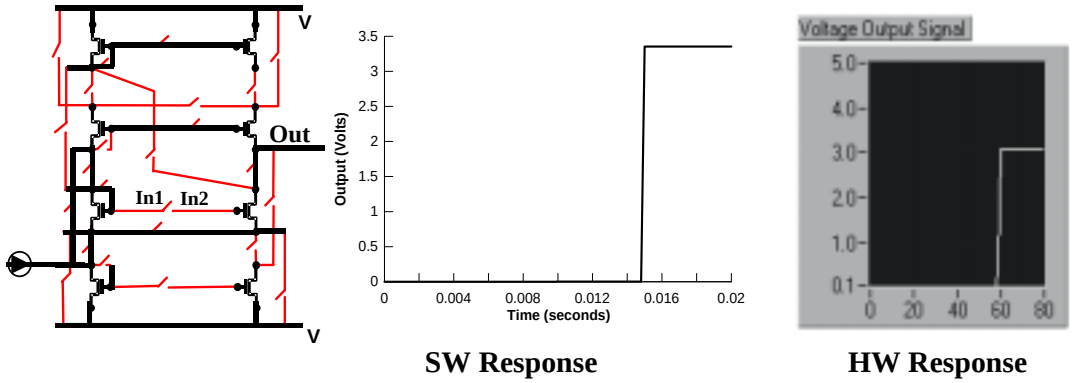


Figure 21 Circuit obtained by mixtrinsic evolution, its valid responses in SW and in HW

6.1.2 Evolution through fuzzy topologies

The technique presented in the following shows accelerated evolution in a number of experiments. Its applicability is tied to the capability of controlling switches in intermediate position (not fully opened, not fully closed) having thus "gray-level" switches with controllable resistance/conductance. Instead of being only ON/OFF, the switches were considered as having a Low/High resistance (Low for ON state). The binary genetic code would thus specify if the switch is Low or High. These switches are implemented through transmission gates (t-gates) and we can control their states to achieve partly opened / partly closed configuration, where they present intermediate resistance. Usually an opened switch corresponds to resistance with a value about 10^8 Ohms, while the closed switch corresponds to a value around 1000 Ohm. The control is achieved with intermediate voltage signals (between 0 V and 5 V for the HP process) applied on the gate of the t-gate switches.

A topology with gray-level switches is named here a fuzzy topology, because it blurs the borders between distinctive circuit topologies: the resulting circuits belong only to certain degrees to fixed (standard) topologies in which two components are either connected or not. This diffuse, fuzzy topology resembles having many seeding topologies simultaneously co-existing, with superimposed effects, the role of evolution being to isolate the most promising topology. In a sense, evaluation of a fuzzy topology is equivalent to simultaneous concurrent evaluation of several superimposed circuit configurations (see Stoica_EH99 for details). Starting evolution with a fuzzy topology with the resistance of the switches being close to an intermediate value between VDD and ground leads quickly to solutions of some value, the convergence towards good solutions continuing simultaneously with a "cooling" effect in which the LOW and HIGH get further apart. This process usually ends with LOW (resistance) becoming ON (fully closed switch) and HIGH becoming OFF, and it had shown one order of magnitude speed-up in simulation of Gaussian circuits [Stoica_EH99].

Faster convergence also means reaching higher fitness values in a given time. This is illustrated by Figure 22 below. Three results are shown: employing switches completely opened or closed; employing partly opened/closed t-gate switches; using switches completely opened or closed and also including ordinary resistors in the simulation models. In the second test we used control voltage values of 1.5 and 3.5 Volts to attain intermediate resistance values. The graph shows the average population fitness over 4 GA executions that sampled 40 individuals along 100 generations [Zebulum_CEC00]. It can be verified that the use of partially opened/closed switches greatly improved the GA performance for this particular problem, since it attained higher fitness values.

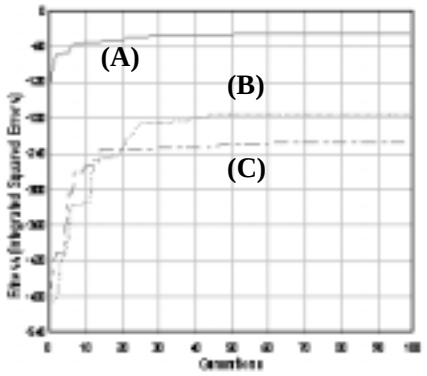


Figure 22 Comparative experiment to assess the effect of gray level switches : (A) – partly opened and closed switches; (B) – completely opened and closed switches; (C) – using resistor model

6.1.3 Design and Reuse

Evolving complex circuits proved hard and a hierarchical approach that is evolving simple circuits first and uses them as building blocks appears natural, yet no successful such experiments are reported in the literature. The method proposed is based upon encapsulation and design re-use and was successfully tested in the evolution of a 4-bit Digital to Analog Converter (DAC). It has been observed that evolution had difficulty in solving this task using low-level components (such as transistors, resistors and capacitors) as building blocks. For instance, the literature reported that a total of 45,000,000 circuits had to be evaluated to achieve the solution for the 3-bit DAC problem using Genetic Programming. The objective of our experiment was therefore to overcome this limitation and synthesize a 4-bit DAC *hierarchically*. At first, a 2-bit DAC was evolved from scratch, e.g., using only MOS transistors as components for evolution. This circuit was then employed as a building block for evolution to synthesize a 3-bit DAC, which was subsequently used as building block for the synthesis of a 4-bit DAC. Figure 23 depicts the evolved circuit schematic and its response. In this particular experiment, the design re-use was not only limited to previously evolved DAC circuits (building blocks), but also employed other well-known building blocks, such as current mirrors.

The experiment took less than one minute in a SPARC Ultra 2 Sun workstation evaluating about 200,000 individuals. The dramatic reduction in time compared to other experiments is due to the fact that the DC operating point of encapsulated building block had already been defined, and their behavior was simulated using a high-level description language (C).

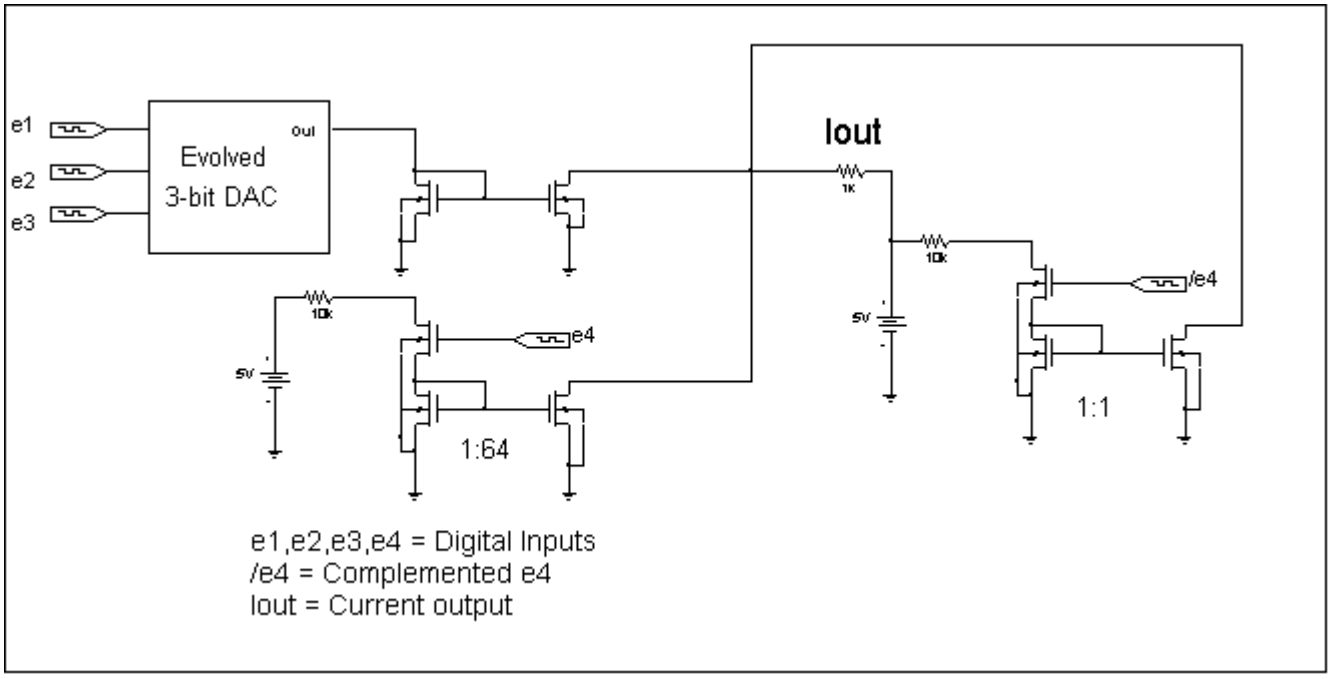


Figure 23 Schematic and circuit response for the 4-bit DAC.

6.2 New Application Directions

6.2.1 Polymorphic electronics

Polymorphic electronics (polytronics) –refers to electronics with superimposed built-in functionality. In this context, a function change does not require switches/reconfiguration as in traditional approaches. Instead, the change comes from modifications in the characteristics of devices involved in the circuit, in response to controls such as temperature, power supply voltage (V_{DD}), control signals, light, etc. The polytronics concept is explained in detail in [Stoica_ICES01]. We present here the results of an experiment encompassing the evolution of a circuit that performs different functions depending on the level of the power supply voltage, V_{DD}. When V_{DD} = 3.3V, the gate behaves as an OR gate; when V_{DD}=1.2V it behaves as an AND gate. An immediate possible application would be to endow circuits with built-in different behavior for *active* or *sleeping* (power saving*)* mode. Figure 24 displays the evolved circuit and plots its response. Polymorphic circuits controlled by temperature or by external signals were also obtained and are described in detail in [Stoica_ICES01]. Polymorphic electronics may have promising applications in fingerprinting/watermarking, information exfiltration or other applications in which it is needed to embed a secret function inside a circuit. Biometric information may also be used embedded as an encoding key.

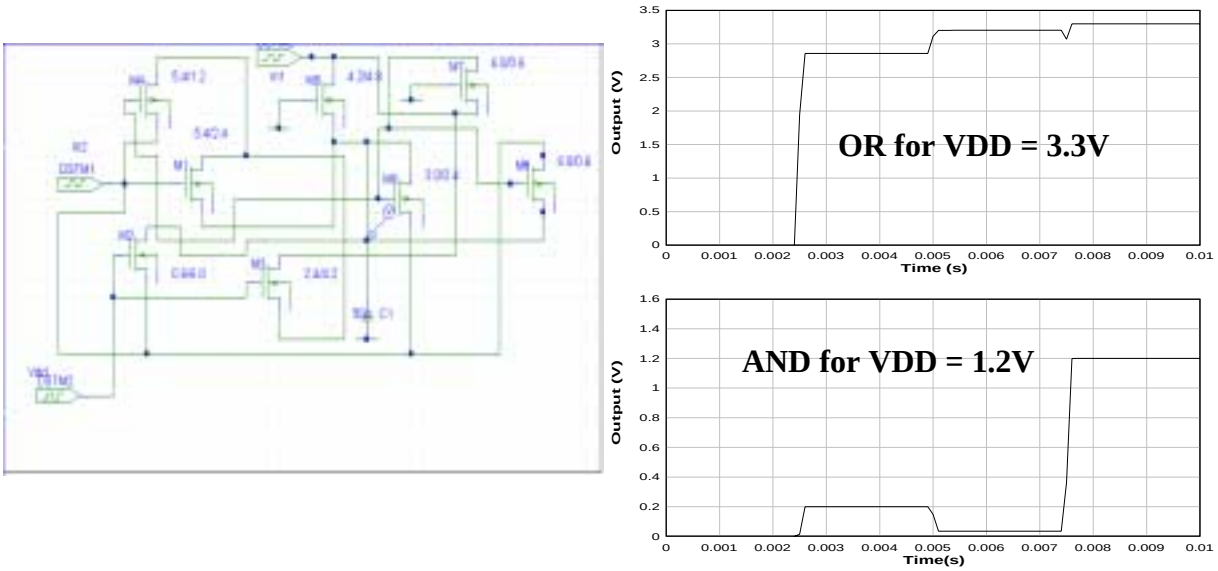


Figure 24 Schematic of the polymorphic circuit controlled by supply voltages (left). In the right, response for two cases, VDD=3.3V and VDD=1.2V. Axis X of the graphs gives the time in seconds.

6.2.2 Evolution/Reconfiguration Based Recovery of Extreme Temperature Electronics

Current approaches seeking expansion of functional range at extreme temperatures focuses on material/device advances. Our Evolvable Hardware approach offers a circuit solution seeking, in-situ, a new configuration on a programmable device, which exploits on-chip devices/components at their temperature-affected operational point. In essence, a new design process takes place automatically, in-situ, under the control of the search algorithm. The configurations could be determined either a priori - part of the original design (which would identify good configurations and store them in a memory) - or in-situ. Once the performance of the current topology starts to deteriorate due to excessive temperature change, the system would switch to a more suitable topology [Stoica_EH01b][Stoica_MAPLD00].

Appendix F describes an experiment whose objective was to recover functionality of the FPTA under extreme temperatures of -197^{o} C and 320^{o} C. The functionality of the FPTA in the experiments were AND, NAND, NOR logic gates. It was verified that evolution was able to successfully recover functionality for these extreme temperatures. One should mention that solutions obtained by evolution at extreme temperature degrade when brought back to room temperature, requiring a new circuit to be configured in the FPTA.

7 Vision and Recommendations for Future Research

Evolvability may become a key characteristic of infrastructures of 2020 and beyond. After the fixed and then the reconfigurable hardware generation, the next one will be self-configurable and evolvable. Evolvable hardware technology will grow at least in three directions: 1) toward including software and evolving as hybrid-ware (the distinction between hardware and software is expected to blur, at least as a far as the schism in system development is concerned), 2) toward including non-electronics, such as antennas, MEMS or biological systems, and 3) toward using parallel evaluations of populations in a network environment. Military devices and systems will have to continuously adapt, automatically, to many simultaneous constraints such as optimal bandwidth allocation, power, jamming-avoidance, humans, threats, mission changes. This is a search/optimization problem in a constantly changing world and evolution can play a major role.

"Smart materials" and distributed, high bandwidth, sensing structures will embed small areas of silicon or another material, containing up to a few hundred programmable transistors or other active devices in areas smaller than 10x10 micron, acting as tiny signal processors co-located with miniature sensing/actuation devices. These would provide local adaptive information processing, for example, in "smart skins" for aircraft or submarines. Or, they could provide soldiers with computational structures seamlessly embedded into his gear.

Certain challenges need to be overcome before this vision becomes a reality. Important ones relate to scaling and complete upfront specifications. The two can be interrelated. So far only relatively simple systems have been evolved. This may be due to the fact that in most cases the components used are primitive elements, for example device-level (transistor, capacitor, resistor) for analog or gate level for digital. For any complex system the number of components used may be relatively large. Considering that number as a required number of components for a solution, the total number of ways of interconnecting them, and consequently the size of the search space where the algorithm will need to look in order to find such a solution is *huge*. Even a simple OpAmp may have 100 transistors, and with three terminals each task of evaluating all combinations is computationally infeasible. The natural solution would be to reduce the space by keeping an acceptable scope of focus. Using higher-level building blocks appears the solution, yet how exactly to select them is still an open problem. The need for upfront complete specifications is reflected in situations when in order to evolve a gate, timing specifications were needed for more than one time domain. Yet, testing devices in a large number of situations for which we want to guarantee its functionality is obviously not efficient.

One way to approach the scalability problem is to first admit that what we address is an open problem for both analog *and* digital (and for system design in general). Many, including us, have been under the impression that for digital the synthesis problem is solved by current techniques and tools. The fact is that only *structural* VHDL (Verilog) is synthesizable, while *behavioral* VHDL is not. The reason is simple: structural offers the problem decomposition! Thus the tools only have to deal with implementation of a simpler block, and also the set of library elements offers easy/direct matches. (The boundary between behavioral and structural depends on the vendor supported language/extension and size of IP library, etc.).

In the context of the above, the following two research directions are recommended for the near future.

a) *Evolutionary-based compilation of behavioral HDL to structural HDL* (analog or/and digital).

b) *Evolutionary-based synthesis of structural AHDL*. Structural AHDL may be the required first step to automatic analog synthesis. The building blocks may be sufficiently small to allow evolution to find optimal solution.

We believe that by decomposing the problem first into a *functional to structural* translation and then a *structural to primitives* one, the chances of evolution to approach complex system are much improved. Another direction we recommend for the near future is:

c) *Hardware/software evolutionary co-design for FPGA/CPU hybrids and other SOC*. This is especially timely in the context of the embedded systems industry rapidly moving toward reconfigurable architectures and devices, with a powerful convergence toward hybrid FPGA/CPU architectures. Ultimately these will use flavors of on-chip hybrids such as the new Xilinx Virtex II Pro chip. There are no tools allowing designers to take advantage of this new computing paradigm. We believe that evolution can do a good partitioning, and an evolution-based tool that would take a system level specification in Matlab/Simulink and convert it to an efficient hardware/software allocation/partitioning.

8 Concluding remarks

Evolution is like a powerful, yet mischievous, genie in a bottle. It will try to do what we ask, yet we should ask carefully since the request can not be changed once formulated: you have to be complete and unambiguous. Otherwise, since the genie will try to minimize his effort, we may get a solution that satisfies our request without being what we really wanted. The scenario says you have to specify how you are going to test the result. Asking for an AND gate specified as correct logical response for combination of input levels at the microsecond scale will get you one, however, the solution may not work at another scale, e.g. at millisecond scale. There is no reason why it should work at another temperature than the one you specified for the test. (Such a thing was not asked/specified). *Formulating the right request is hard*. In many respects *evolutionary engineering will shift the scope from designing systems to designing system specifications and verification procedures* (fitness functions). If the right question is asked a correct response will be obtained. Evolution is powerful, and works without design domain knowledge. To fully learn how to deal with the genie we needed tools to experiment with, and have opportunity to ask in many ways to see how it behaves. This effort has created the technology base that enables users to evolve real hardware in seconds. This is very important since it is an enabler for future research and development of this technology, eliminating days and weeks of waiting for results of a simulation experiment. Not only hardware evaluations are fast so solutions can be obtained in much less time than in simulations: the solutions are also guarantied to work since they were already tested during the evolution directly in hardware. Simulations may still have advantages in some situations, for example if we need to exercise a solution to a set of conditions not convenient for hardware, such as a set of extreme temperatures, for the purpose of building a wide range robust circuit. The required testbed would not be fast to build, cheap or convenient. Thus, a simulated solution may easily change the conditions and run an evaluation test. On the other hand we may not have good models for some of the behaviors such as at extreme temperature. In this case evolution with real hardware is the only option. Evolution also goes well hand in hand with the industry trend of building faster devices, operating at higher frequencies, since this also enables faster response evaluations, and thus overall evolution faster. If however the specifications are for a slow response constant (e.g. controller timers for car wiper blades) evaluation in software would be probably faster since we could "artificially accelerate" evaluations. Simulation offers easier "what-if" testing means.

Where and how to call the genie? For adaptive devices hardware evolution is essential. Special concerns relate with possible limitations of the number of reconfigurations allowed by the device, with safety at system level if new configurations may produce dangerous situations, such as if it evolves a helicopter stabilizing controller during a real flight. Evolving sensors may be less risky than evolving controllers. In any case, evolvable hardware has arrived, and will shape a new generation of hardware in terms of flexibility and survivability. The first generation was fixed hardware. The second generation was reconfigurable hardware. The third generation will be self-configurable/evolvable hardware. We predict that infrastructures of tomorrow will be evolvable, but who will have the rights to make the wishes for the genie, and how would that affect us all? Evolution can be used for anti-virus creation, but can be used also for creating new adaptive and morphing viruses. The programs that implement the genie may in the future subject to self-improvement and changing themselves. And, what if they change in a more "convenient" way and instead of listening to our requests will start formulating their own, which may be against our interests? In other words, what if the genie goes out of the bottle?

9 Reference

Books

[Zebulum_CRC02] R. Zebulum, M. Pacheco, M. Vellasco. Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. CRC Press, 2002.

[Keymeulen_IEEEProc01] D. Keymeulen, A. Stoica, J. Lohn, R. Zebulum (eds.). Proceedings of the Third NASA/DoD Workshop on Evolvable Hardware. Long Beach, CA, July 12-14, 2001. IEEE Computer Society Press.

[Lohn_IEEEProc00] J. Lohn, A. Stoica, D. Keymeulen, S. Colombano (eds.). Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware. Palo, Alta, CA, July 13-15, 2000. IEE Computer Press.

[Stoica_IEEEProc99] A. Stoica, D. Keymeulen and J. Lohn (eds.). Proceedings of the First NASA/DoD Workshop on Evolvable Hardware. Los Alamitos, CA 90720, July 1999. IEEE Computer Society Press.

Journals

[Stoica_IEEEJour01] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, Reconfigurable VLSI Architectures for Evolvable Hardware: from Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips. In IEEE Transactions on VLSI Systems, Special Issue on Reconfigurable and Adaptive VLSI Systems, vol. 9, No. 1,February 2001. (pp.227-232).

[Keymeulen_IEEEJour00] D. Keymeulen, A. Stoica, R. Zebulum. Fault-Tolerant Evolvable Hardware using Field Programmable Transistor Arrays. In IEEE Transactions on Reliability, Special Issue on Fault-Tolerant VLSI Systems, vol. 49, No. 3, 2000 September. (pp.305-316) IEEE Press.

[Zebulum_MITJour00] Zebulum, R.S, Pacheco, M.A., Vellasco, M., Variable Length Representation in Evolutionary Electronics. In Evolutionary Computation Journal, MIT Press, (pp. 93-120), vol.8, No.1, Spring, 2000.

[Zebulum_SBMICROJour00] Zebulum, R.S, Pacheco, M.A., Vellasco, M., A Novel Multi-Objective Optimization Methodology Applied to the Synthesis of CMOS Operational Amplifiers. In Journal of Solid-State Devices and Circuits, Microelectronics Society - SBMICRO, ISSN 01049631, vol. 8, N. 1, (pp. 10-15), edited by Wilhelmus Van Noije, 2000, February 2000.

[Thompson_IEEEJour99] Thompson, A., Layzell, P., Zebulum, R.S., "Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution", published at: IEEE Transactions on Evolutionary Computation, Special Issue on Evolvable Hardware, Moshe Sipper (Ed), (pp.167-196), vol. 3, N.3, September, 1999.

Conferences

[Ferguson_GECCO02] Ferguson, M.I., Stoica A., Zebulum R., Keymeulen D. and Duong, V. "An Evolvable Hardware Platform based on DSP and FPTA". To be published in Proceedings of the Genetic and Evolutionary Computation Conference, July 9-13, 2002, New York, New York.

[Zebulum_IEEEAero02] Zebulum, R., Stoica, A., Keymeulen, D., Ferguson, M.I., Duong, V., Daud, T. "Current Mode Circuits: Increasing the Chances of Evolution to Find a Way". In Proceedings of the IEEE Aerospace Conference, 9-16 March 2002, Big Sky, Montana. New York: IEEE Press.

[Buehler_IEEEAero02] M. Buehler, G. Kuhlman, D. Keymeulen and S. Kounaves. "Advanced Electronic Tongue Concept". In Proceedings of the IEEE Aerospace Conference, 9-16 March 2002, Big Sky, Montana. New York: IEEE Press.

[Stoica_ICES01] Stoica, A. Zebulum R. and Keymeulen D. "Polymorphic Electronics" International Conference on Evolvable Systems, October 2001, Tokyo, Japan. (pp. 291-302).

[Keymeulen_ICES01] Keymeulen D., Zebulum R., Stoica A. and Buehler M. “Initial Experiments of Reconfigurable Sensor Adapted by Evolution”. International Conference on Evolvable Systems, October 2001, Tokyo, Japan. (pp.303-313).

[Stoica_EH01a] A. Stoica, R. Zebulum, and D. Keymeulen, “Progress and Challenges in Building Evolvable Devices”, Third NASA/DoD Workshop on Evolvable Hardware, Long Beach, July, 12-14, 2001, (pp.33-35), IEEE Computer Society.

[Stoica_EH01b] A. Stoica, D. Keymeulen, and R. Zebulum, “Evolvable Hardware Solutions for Extreme Temperature Electronics”, Third NASA/DoD Workshop on Evolvable Hardware, Long Beach, July, 12-14, 2001, (pp.93-97), IEEE Computer Society.

[Keymeulen_GECCO01] D. Keymeulen, A. Stoica, M. Beuhler, R. Zebulum, V. Duong. “Evolutionary Mechanisms for Smart On Board Adaptive Sensing Applied to the MECA Electrometer”. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). (pp. 1197-1204). July 7-11 2001, USA, Menlo Park, CA: AAAI Press.

[Keymeulen_IEEEAero01] D. Keymeulen, A. Stoica, M. Buehler, R. Zebulum, V. Duong. “Evolutionary Mechanics for Smart On-Board Adaptive Sensing applied to MECA Electrometer”. In Proceedings of IEEE Aerospace Conference, March 10-17, 2001, Big Sky, Montana, USA. Manhattan Beach, CA. IEEE Press. (Published in CD)

[Zebulum_IEEEAero01] R.Zebulum, A. Stoica, D. Keymeulen, "Experiments on the Evolution of Digital to Analog Converters", published in the Proceedings of the 2001 IEEE Aerospace Conference, March 2001, Montana. ISBN: 0-78-3-6600-X. (Published in CD)

[Stoica_NASA01] Stoica, A. Thakoor, D. Keymeulen and R. Zebulum, T. Daud and B. Toomarian. “Evolvable Hardware for Extreme Environments: Hot or Cold, We Live Long”. In Forum on Innovative Approaches to Outer Planetary Exploration, February 21-23, 2001, Houston, Texas.

[Keymeulen_ANNIE00] D. Keymeulen, A. Stoica, R. Zebulum, G. Klimeck, C. Lazarzo, Y. Jin. “EHWPack: An Evolvable Hardware Environment Using the Spice Simulator and the Field of Programmable Transistor Array”. In the Proceedings of ANNIE 2000 (Smart Engineering System Design) St Louis, MO, November 5-8, 2000

[Stoica_ANNIE00] A. Stoica, R. Zebulum, D. Keymeulen, Y. Jin, T. Daud, “Evolutionary Design of Smart Analog Circuits”. In the Proceedings of ANNIE2000 (Smart Engineering System Design), St. Louis, MO, pp. 245-250. ASME press. November 5-8, 2000.

[Keymeulen_IRW00] D. Keymeulen, A. Stoica, R. Zebulum, Y.Jin, V. Duong. “Fault-Tolerant Approached based on Evolvable Hardware and Using a Reconfigurable Electronic Device”. In Proceedings of the IEEE International Integrated Reliability Workshop, October 23-26, 2000, Lake Tahoe, CA, USA. Manhattan Beach, CA. IEEE Press. (pp.32-39)

[Stoica_MAPLD00] A. Stoica, D. Keymeulen, R. Zebulum, Y. Jin and V. Duong, “Evolvable Hardware for Extreme Environments: Expanding Device Operational Envelope through Adaptive Reconfiguration”, The 3rd annual Military and Aerospace Applications of Programmable Devices and Technologies International Conference, 2000 MAPLD International Conference, JHU/APL, Laurel, MD September 26-28, 2000.

[Stoica_HR00] Stoica A. “Robot Fostering Techniques for Sensory-Motor Development of Humanoid Robots” First International Conference on Humanoid Robots, Sept 7-8, 2000, MIT.

[Keymeulen_CEC00] D. Keymeulen, A. Stoica, R. Zebulum, V. Duong. “Results on the Fitness and Population based Fault Tolerant Approaches using a Reconfigurable Electronic Device”. In Ali Zalzala (eds.), Proceedings of the Congress on Evolutionary Computation (CEC-2000), (pg. 537-544). July 16-19, 2000, La Jolla, USA. Manhattan Beach, IEEE Press.http://ehw.jpl.nasa.gov/Documents/PDFs/CEC2000Didier.pdf

[Zebulum_CEC00] R. Zebulum, A. Stoica, D. Keymeulen. “Design process of an evolutionary oriented reconfigurable architecture”. In Proceedings of the 2000 Congress on Evolutionary Computation (CEC2000), July 16-19, 2000, San Diego, USA. Manhattan Beach, CA, (pp. 529-536). IEEE Press, n. 00TH8112.

[Stoica_EH00] A. Stoica, D. Keymeulen, A. Thakoor, T. Daud, G. Klimech, Y. Jin, R. Tawel, V. Duong. “Evolution of Analog Circuits on field Programmable Transistor Arrays”. In J Lohn et al. (eds.), Proceedings of NASA/DoD Workshop on Evolvable Hardware (EH2000), July 13-15,2000, (pp.99-108). Palo Alta, CA, USA. IEEE Computer Society.

[Keymeulen_GECCO00] D. Keymeulen, G. Klimeck, R. Zebulum, A. Stoica, and C. Lazaro. “EHWPack: A Parallel Software/Hardware Environment for Evolvable Hardware”. In Whitley Darrell (eds.), Proceedings of the Genetics and Evolutionary Computation Conference (GECCO-2000), July 8-12, 2000,pg.538-539. Las Vegas, Nevada USA. San Francisco, CA: Morgan Kaufmann.

[Stoica_IEEELogic00] A. Stoica. “Evolvable Hardware: From On-Chip Circuit Synthesis to Evolvable Space Systems”. In Proceedings of the 30th IEEE Symposium on multi-valued logic, May 23-25, 2000, Portland, USA, IEEE Press.

[Stoica_ICES00] A. Stoica, R. Zebulum and D. Keymeulen. “Mixtrinsic Evolution”. In T. Fogarty, J. Miller, A. Thompson and P. Thompson, (eds.), In Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware. (ICES2000). April 17-19, 2000, Edinburgh, UK. New York, USA, Springer Verlag. (pg.208-217)

[Zebulum_ICES00] R. Zebulum, A. Stoica and D. Keymeulen. “A Flexible Model of a CMOS Field Programmable Transistor Array Targeted for Hardware Evolution”. In T. Fogarty. J. Miller, A. Thompson and P. Thompson, (eds.), In Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware. (ICES2000) April 17-19, 2000, Edinburgh, UK. New York, USA, Springer Verlag.

[Stoica_IEEEAero00] A. Stoica, D. Keymeulen, V. Duong and C. Lazaro. “Automatic Synthesis and Fault-Tolerant Experiments on an Evolvable Hardware Platform”. In R. Profet, D. Woerner and R. Wright, (eds.), Proceedings of IEEE Aerospace Conference, March 18-25, 2000, Big Sky, Montana, USA. Manhattan Beach, CA: IEEE Press.

[Stoica_CRC99] Stoica, A. “Learning eye-arm coordination using neural and fuzzy neural techniques”. In H.N Teodorescu, A. Kandel, L. Jain, (Eds.) Soft Computing in Human-Related Sciences, CRC Press, 1999, pp. 31-61.

[Stoica_EH99] A. Stoica, D. Keymeulen, R. Tawel, C. Lazaro and Wei-te Li. “Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits”. In Stoica A., Keymeulen D. and Jason L. (eds.) Proceedings of the First NASA/DoD Workshop on Evolvable Hardware (EH-1999), (pg.76-84). July 19-21, 1999, Pasadena, California USA. Los Alamitos, CA 90720: IEEE Computer Society Press.

[Zebulum_EH99] Zebulum, R. S., Pacheco, M., Vellasco, M., “Artificial Evolution of Active Filters”, Proceedings of First NASA/DoD Workshop on Evolvable Hardware (EH-1999), (pp. 66-75) IEEE Computer Society press, ISBN 0-7695-0256-3.

[Stoica_GECCO99] A. Stoica, C. Lazaro, D. Keymeulen and K. Hayworth. “Evolution of CMOS Circuits in Simulation and Directly in Hardware on a Programmable Chip”. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M. and Smith, R.E. (eds.), GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, (pp. 1198-1203). July 13-17, 1999, Orlando, Florida USA. San Francisco, CA: Morgan Kaufmann.

[Stoica_CEC99] A. Stoica, G. Klimeck, C. Lazaro, D. Keymeulen and A. Thakoor. “Evolutionary Design of Electronic Devices and Circuits”. In Fogel D. and Schoenauer M. (eds.) CEC99: Proceedings of 1999 Congress on Evolutionary Computation, (pp. 1271-1278). July 6-9, 1999, Washington DC USA. Piscataway, NJ USA, IEEE Press.

[Stoica_FUSION99] A. Stoica, Wei-te Li, T. Thomas, T. Daud, and J. Fabunmi (1999) “Extended Logic Intelligent Processing System as a Sensor Fusion Processor Hardware”, The 2nd International Conference on Information Fusion, FUSION'99, pg. 611-618. July 6- 8, Sunnyvale, CA, (pp. 611-618).

[Mandutianu_ICAI99] S. Mandutianu and A. Stoica “An Evolvable Multi-agent Approach to Space Operations Engineering” .The 1999 International Conference on Artificial Intelligence, IC-AI'99, Las Vegas, NV, June 28 - July 1, 1999

[Stoica_MICROEURO99] A. Stoica. “Towards Evolvable Hardware Chips: Experiments with a Programmable Transistor Array”. Proceedings of the 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-inspired Systems”, In Microneuro’99, Granada, Spain, April 7-9, 1999, IEEE Computer Society (pp. 156-162).

[Stoica_IEEEAero99] A. Stoica, D. Keymeulen, C. Lazaro, K. Hayworth and R. Tawel. “Towards On-board Synthesis and Adaptation of Electronic Functions: An Evolvable Hardware Approach”. In R. Profet, D. Woerner and R. Wright, (eds.), Proceedings of IEEE Aerospace Conference, March 6-13, 1999, Snowmass, Colorado USA. Manhattan Beach, CA: IEEE Press.

[Stoica_ICES98] A. Stoica, A. Fukunaga, K. Hayworth, and C. Salazar-Lazaro. (1998) “Evolvable Hardware for Space Applications”. In M. Sipper, D. Mange and A. Perez-Uribe (Eds.) Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science, Springer, Berlin, (pp 166-173)

[Zebulum_SEAL98] Zebulum, R.S., Pacheco, M.A., Vellasco, M., “Evolutionary Systems Applied to the Synthesis of a CPU Controller”, Second Asia-Pacific Conference on Simulated Evolution and Learning, Canberra, SEAL98, Australia, Novembro, Lecture Notes in Artificial Intelligence, Springer Verlag, 1998.

[Klimeck_ICSDT98] Gerhard Klimeck, Chris Bowen, Tim Boykin, Fabiano Oyafuso, Tom Cwik, Carlos Salazar-Lazaro, and Adrian Stoica “The Nanoelectronic Modeling Tool NEMO and its extension to High Performance Computing”, ICSDT 98 - 6th International Conference on Simulation of Devices and Technologies, Cape Town, South Africa, Oct 14-16, 1998.

[Stoica_MAPLD98] Stoica, A, Carlos-Salazar Lazaro, Raoul Tawel “Evolvable Electronic Systems”. In 1998 Military and Aerospace Applications of Programmable Devices and Technologies Conference MAPLD’98, NASA Goddard Space Flight Center, Sept 15-16, 1998

[Zebulum_SBCCI98] Zebulum, R.S., Pacheco, M.A., Vellasco, M., “Synthesis of CMOS Operational Amplifiers Through Genetic Algorithms”, Proceedings of the SBCCI98 (Brazilian Symposium on Integrated Circuits), ISBN 0-8186-5, pp. 125-128, Rio de Janeiro, Brazil, September, 1998.

[Klimeck_DARPA98] Gerhard Klimeck, Carlos H. Salazar-Lazaro, Adrian Stoica, and Tom Cwik, “Structural Analysis Using Quantum Mechanical Electron Transport Simulations Driven by a Genetic Algorithm”, Second Workshop on Characterization, Future opportunities and Applications of 6.1Å III-V Semiconductors, Organized by ONR, NRL, and DARPA, Naval Research Laboratory, Washington DC, August 24-26, 1998.

[Zebulum_ICEC98] Zebulum, R.S., Pacheco, M.A., Vellasco, M., “Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design”, IEEE International World Congress on Computational Intelligence, ICEC’98, pp.434-439, Alaska, May 1998.

[Fukunaga_IEEEAero98] Fukunaga, K. Hayworth, A. Stoica, (1998) “Evolvable Hardware for Spacecraft Autonomy”, IEEE Aerospace Conference, Snowmass CO, March 1998

[Wingate_MS97] Wingate, M. and Stoica, A. (1997) “An Application of Fuzzy Neurons to Visual Learning”. In International Conference on Modeling and Simulation MS'97, Melbourne, Australia, 29-31 Oct., (pp. 45-50).

[Stoica_ISS97] Stoica, A. “On hardware evolvability and levels of granularity”. Proc. of the International Conference on Intelligent Systems and Semiotics 97: A Learning Perspective, NIST, Gaithersburg, MD, Sept. 22-25, 1997.

[Stoica_ISIS97] Stoica, A. and Blosiu, J. “Neural learning using Orthogonal Arrays”. Proc. of the International Symposium on Intelligent Systems AMSE-ISIS’97, Reggio Calabria, Italy, Sept. 11-13, 1997.

[Zebulum_IEEEMIDWEST96] Zebulum, R.S., Perelmuter, G., Vellasco, M., Pacheco, M.A., “A Comparison of Different Spectral Analysis Models for Speech Recognition Using Neural Networks”, IEEE MIDWEST Conference on Circuits and Systems, Iowa, USA, August 1996.

Manuals

[INNOVATIVE_M1] “SBC6x Development Package Manual”, Innovative Integration ©2001, (805)-520-3300.

[INNOVATIVE_M2] “OMNIBUS User’s Manual”, p255-286, Innovative Integration, © 2001, (805)-520-3300.

[INNOVATIVE_M3] “OMNIBUS User’s Manual”, p143-152, Innovative Integration, © 2001, (805)-520-3300.

Reports

[Stoica_TEMP01] “Evolvable Hardware Enhance Survivability for Extreme Temperature”, A. Stoica, D. Keymeulen, N. Mehta, V. Duong, R. Zebulum and I. Ferguson. Jet Propulsion Laboratory, ©2001.

[Stoica_EHWPACK01] EHWPack Development System for Spice Software and FPTA Hardware User and Developer Manual, EHW Group, JPL intern document. A.Stoica, G. Klimeck, D. Keymeulen, R. Zebulum and C. Lazaro.

[Stoica_FPTA2_DIG_02] FPTA2 Digital Part Documentation, JPL intern document. A.Stoica, J. Gilbert, D. Keymeulen, R. Zebulum , M.I. Ferguson, V. Duong

[Stoica_FPTA2_ANA_02] FPTA2 Digital Part Documentation, JPL intern document. A.Stoica, R. Zebulum, D. Keymeulen, R. Zebulum , M.I. Ferguson, V. Duong

WebPages

[EHW_WEB] ehw.jpl.nasa.gov, webpage of the Evolvable Hardware Research Group at the Jet Propulsion Laboratory.

Computer Code

[Ferguson_GA1] “ga1.c”, Computer Program, M. I. Ferguson, Jet Propulsion Laboratory, ©2001.

Appendix A Objective and Statement of Work: proposed and accomplishment.

Objective

The Jet Propulsion Laboratory (JPL) will develop and demonstrate self-reconfigurable circuits that evolve directly in hardware on a VLSI chip. The evolved circuits will perform complex signal processing functions, such as adaptive filtering or randomization. This development will break new ground and demonstrate a new paradigm for the design of adaptive computing, i.e. evolvable hardware. It will shape new tools and approaches, and it will also establish a technology base that others can use.

We consider that the objective has been achieved. We are able to evolve, directly in hardware on reconfigurable VLSI chip circuits of adaptive functionality. Their complexity is somehow less than expected, yet the concept is demonstrated, and is now a matter of extra effort in techniques for approaching complex systems, including the use of hierarchy.

Statement of Work (SOW)

The Jet Propulsion Laboratory (JPL) will develop and demonstrate a self-reconfigurable, evolvable hardware chip for the Defense Advanced Research Projects Agency. In the performance of this work, JPL will, on a best effort basis:

1. Design and perform experiments in evolution of analog and digital computational circuits using a reconfigurable Programmable Transistor Array architecture.

Evolved analog computational circuits (gaussian, cubic, fuzzy t-norms, max/min, multiplier, etc) and digital gates using the PTA architecture, in various conditions including extreme temperatures.

2. Co-design an improved, enhanced PTA and a genetic search processor and perform evolutionary experiments with the simulated enhanced PTA, evolving signal processing circuits, such as adaptive filters and randomizers.

Designed several versions of PTA, performed optimization studies on the genetic processor, evolved adaptive filters and computational circuits.

3. Define, design, and fabricate a Genetic Processor Chip and demonstrate its functionality.

Implemented and demonstrated the Genetic Processor on a commercial DSP chip.

4. Develop an Evolvable Hardware Testbed, including data acquisition and signal processing boards, test boards and a graphical user interface for evolutionary experiments.

Developed the EH Testbed allowing evolution in software, in hardware, and in mixed mode (HW/SW). Uses various data acquisition boards, graphical interfaces, interconnection with supercomputer.

5. Define, design, fabricate an evolvable chip.

Designed, fabricated and demonstrated three generations of evolution-oriented chips, of which the latest (FPTAs) forms a board level stand-alone evolvable system with a DSP chip.

6. Demonstrate to DARPA, at JPL site, on-chip evolution for an electronic functionality from the category of applications of interest to DARPA, such as adaptive interference canceling or adaptive transfer function matching.

April 1st 2002 demonstration.

7. Prepare documentation of the effort including the chip and experimental results.

This document, CD, web-based documentation and papers.

8. Attend meetings as required by DARPA

Attended all DARPA PI meetings.

9. Prepare periodic reports and a final report.

Periodic reports were given in PowerPoint format at each DARPA PI meeting. This document (and associated CD) constitutes the final report.

Appendix B Documentation for FPTA chip

FPTA2 – Analog Part Documentation

FPTA2 – Digital Part Documentation

These two documents describe respectively the architecture of the analog and digital parts of the FPTA 2 chip.

The analog part documentation provides a description of the array of re-configurable cells and also presents a broader view of the FPTA-2 chip. The document is composed of 16 sections. *Section 1* introduces the basic features of the chip, e.g., technology, number of transistors, resistors, capacitors and the die size. It also provides other general information, such as packaging and reconfiguration time. In *Section 2*, the basic cell structure is described in terms of a block diagram. In *Section 3* it is shown the pattern of interconnection between cells, which are classified as inner, vertex and boundary cells according to their position in the array. *Sections 4 to 7* of the document describe the cell sub-blocks introduced in Section 2, the photo detectors, analog memory, re-configurable resistances and re-configurable circuitry. Particularly, *Section 7* presents the re-configurable circuitry, which is the core of the cell. It shows the schematic for this sub-block, as well as examples of analog building blocks that can be mapped onto the re-configurable circuitry. Other specific information on transistors and capacitor sizes are also included in this section. *Section 8* depicts a diagram with the overall cell array, the cell interconnections and the cell-numbering scheme adopted for programming purposes. Section 9 describes the addressing mechanism used to program the cells. *Section 10* classify the 256 pads used in the chip according to their functionality, e.g., data bus, address bus, power pins, analog control signals, analog inputs, analog outputs, etc. *Section 11* presents some experiments performed during the FPTA2 design process. Particularly, it presents the mapping onto the FPTA-2 cell, and respective simulation, of Operational Amplifier, Gaussian circuit, common-source amplifier and photo-detecting circuits. *Section 12* provides a more detailed schematic of the cell, including the list of all internal/external signals coming to and leaving from the cell, which served as a guide during the schematic driven layout process performed during the chip design. It is also depicted the connections of resistance/capacitive resources to the re-configurable circuitry. *Section 13 and 14* respectively describe the analog inputs and outputs numbering and association to the chip pins. *Section 15* presents some analog limitations considered during the design, such as maximum operating frequency, maximum current and loading. Finally *Section 16* summarizes the analog I/O interconnections, going from a board level description to the cell schematic.

The documentation of the digital part describes the procedure used to program the cell array and the control logic circuits, including the *write* and *reset* logic, and the *cell control logic*. The document starts by listing all internal and external chip control signals. Next the *write protocol* to program the re-configurable cell array is described, including a timing diagram and a diagram of the digital circuit.

Similarly, the *Reset* control logic and circuitry is presented afterwards. Following the cell control logic is presented, referring to the array of latches inside each cell, which determine the state of the switches. A timing diagram is provided as well. The documentation also provides simulation results of this previously mentioned latch components used in the cell control logic, showing setup time limitations. Finally, a block diagram of the clock network design concept is presented.

Evolutionary Processor

The evolutionary processor is a collection of architecture independent routines to perform tasks related to evolution in hardware. These include the baseline genetic algorithm functions (sort, crossover, mutation) as well as the I/O functions related to stimulation and evaluation of the individuals.

As stated above, the genetic algorithm (GA) iterates over a genome consisting of candidate configurations, making modifications between subsequent generations. The genome is a population P of N chromosomes each containing a fitness value, and a set of genes corresponding to physical cell structures. An example structure can be found in Figure 25, where each gene consists of 80 configuration bits, which are split into five 16-bit words.

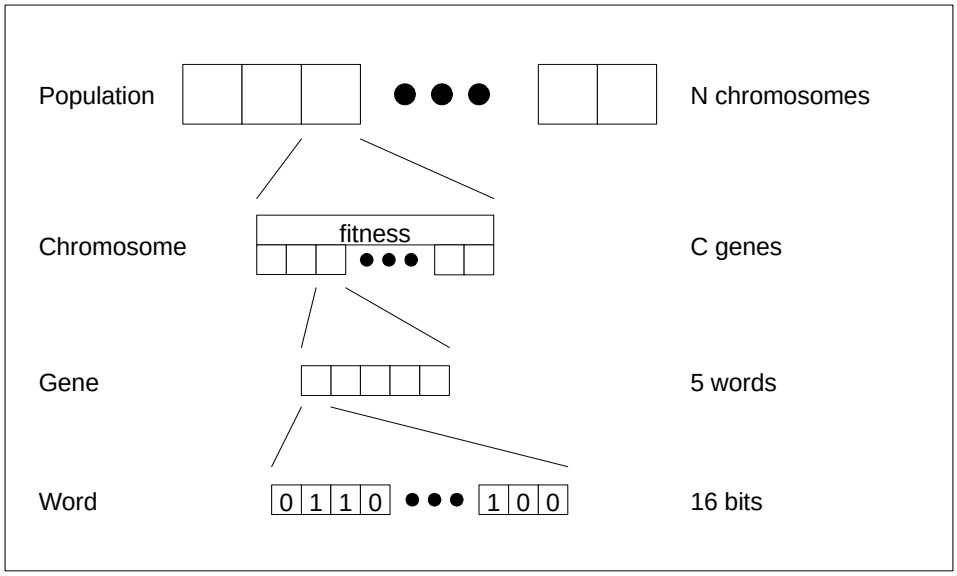


Figure 25 An example memory structure of a population of candidate solutions. Each individual has a fitness and configuration data. The configuration data is broken into genes and further subdivided into five 16-bit words, which correspond to the 80 configuration bits.

The genetic algorithm employed in version 1.0 of the evolutionary code is described in pseudo-code below.

Inputs:

- P the population
- t_s sample time
- $R(t)$ the current response of the chip
- $W(t)$ the desired response of the chip
- $S(t)$ a stimulus to use
- n the number of samples
- N the population size
- F_t the target (minimum acceptable) fitness
- C Number of genes to use (how much of the chip to utilize)
- P_c Crossover probability
- P_m Mutation probability
- P_e Elite percentage to save

Outputs:

F_0 the fitness of best individual for each generation
P_0 the final chromosome corresponding to the solution (chip configuration)

Algorithm:

```
Initialize(P,N)
do{
  for(i=0;N){
    Program P(i)
    Stimulate the individual S(t_s)
    Evaluate the response (W(t),R(t))
  }
  P_0=Sort(P)
  Select elite individuals(P)
  Crossover(P)
  mutation(P)
}until (F_0•F_t)
return(P_0)
```

Stimulate/Response: The stimulation of the individuals can be accomplished by several means but in all cases the algorithm triggers when the stimulus/response cycle begins.

Fitness evaluation: Each individual is evaluated based on criteria input by the user in the form of a function, which returns a single value corresponding to how fit the individual is. This is the metric by which all individuals in the population are ranked. A typical fitness function will be a sum over all samples of some equation comparing the actual response, $R(t)$, to a desired one, $W(t)$, for example:

$$F = \sum_{t_s=0}^{n-1} f(t)(R(t) - W(t))^b$$

where $f(t_s)$ is some function and b is some parameter. This function may be as complex as the user desires, but must return a single value for ranking purposes.

Sort: The sort function is implemented as the standard quick sort algorithm, sorting from low to high values of fitness. Since the population is implemented as an array of pointers, the memory overhead for small populations is not prohibitively expensive.

Crossover: The crossover function selects some percentage of the individuals and for those individuals selects two bit indices and swaps the sections between those two indices between two individuals. The selection of the two individuals to cross over is based on fitness. This is generally known as the wheel method because it normalizes the sum of all selection probabilities to 1 and then places each individual into a range of probability between 0 and 1, with the more fit individuals getting a larger fraction of the available probability space. A random real number is then chosen between 0 and 1 and the corresponding individual is chosen. This allows that more fit individuals get selected for crossover more often.

The actual crossover of two individuals is done by considering the two individuals as a configuration bit streams and by selecting a start and a stop index (called *two-point crossover*) and by swapping the bits from the two individuals. No consideration is given to gene structure during this operation.

Mutation: The mutation function essentially loops a number of times corresponding to $P_m \times N$ and on each pass chooses an index into the configuration bit stream and flips it. This does not enforce that the same bit doesn't get flipped twice, but in doing so the function runs in O(n) time rather than $O(n^2)$.

Appendix C Stand-alone Board-level Evolvable System

This section refers to a stand-alone DSP and board-level evolvable system onto which we have implemented the EP structure.

The EHW-DSP testbed consists of a PC connected through JTAG to an Innovative Integration SBC67 stand-alone DSP board [INNOVATIVE_M1]. This DSP is also connected to the FPTA2 motherboard designed at JPL as seen in Figure 27. The SBC67 has a Texas Instruments (TI) TMS320C6701 floating-point processor with 128KB internal SRAM and 16MB of external SRAM. It also has two add-on modules connected through a proprietary OMNIBUS interface, a SERVO-16 [INNOVATIVE_M2] providing 16 analog input and output with 16-bits of precision and simultaneous sampling at 100kSample/sec and a DIG-32 [INNOVATIVE_M3] for 32 additional digital I/O operating at 7.5Mhz.

The evolutionary algorithms (Appendix C) are developed on the PC using the TI Code Composer environment. The compiled code is then downloaded to the DSP via a JTAG connection. The JTAG connection is used subsequently to monitor and control the algorithm pending development of a host application, which can communicate more efficiently with the DSP via USB

The evolutionary algorithm is executed on the DSP and communicates with the FPTA digital interface through both the dedicated 32-bit interface on the SBC67 as well as the DIG-32 Omnibus module. The DSP system can be seen in Figure 27. As shown in the figure, there are two paths from the DSP to the FPTA, the digital path along which the candidate configurations are downloaded and an analog path which is used when stimulating the FPTA and recording the response for evaluation on the DSP.

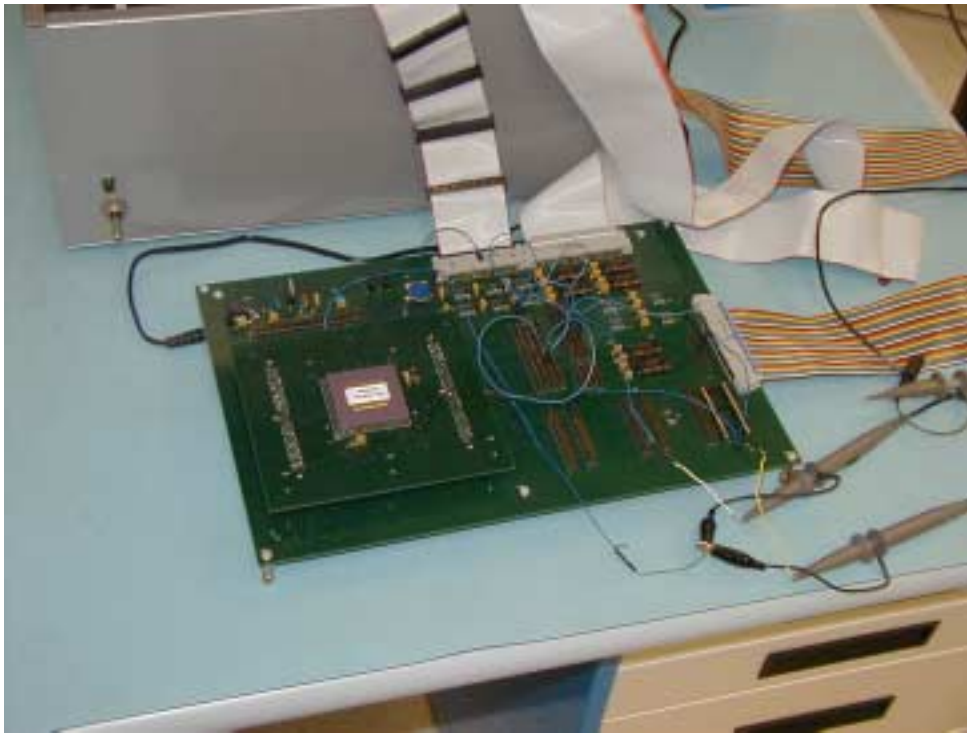


Figure 26 A picture of the Stand-Alone Board-Level Evolvable (SABLE) System.

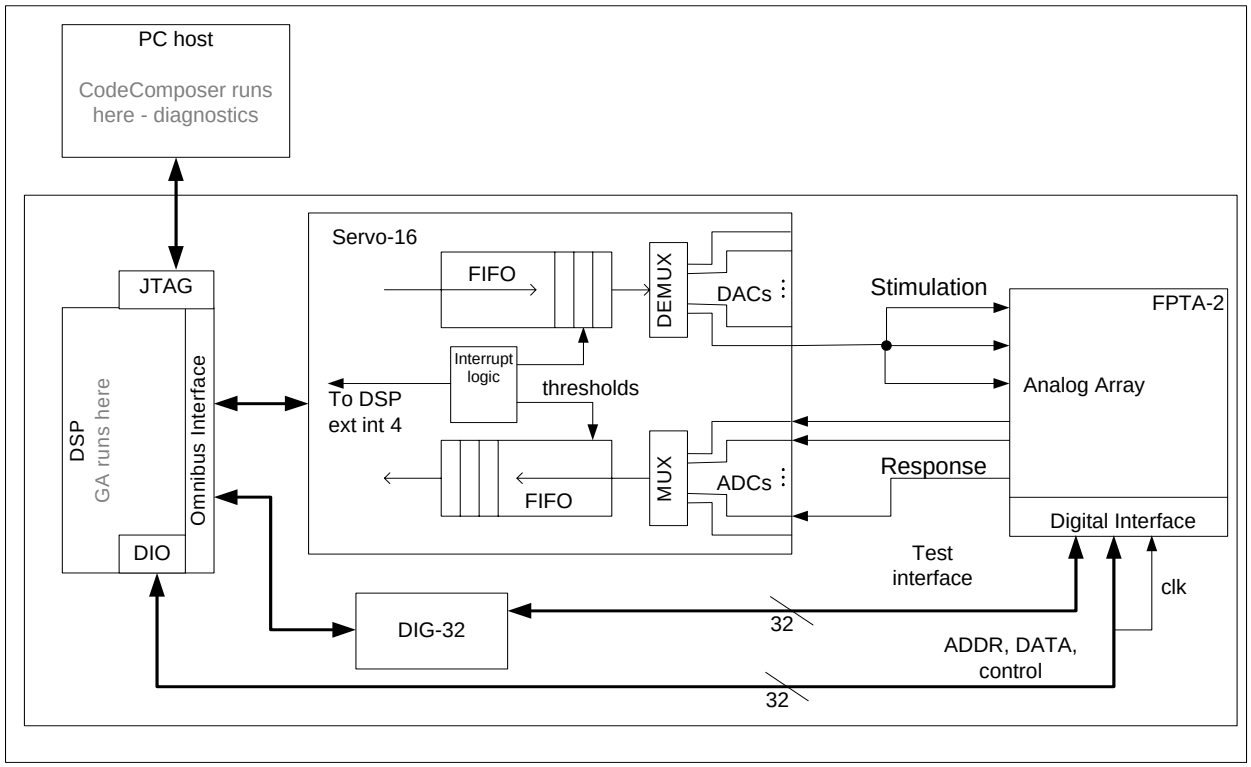


Figure 27 The structure of the stand-alone Evolvable System. The PC host is outside the stand-along environment and is used for diagnostic purposes.

Software – Architecture Dependent:

The software implementing the genetic algorithm is designed around a protocol called DSP/BIOS designed by TI. DSP/BIOS is a real-time kernel, which allows task scheduling and synchronization as well as configuration of the software architecture and advanced debugging features. The task scheduling ability is used to configure software as well as hardware interrupts. These interrupt routines can interact in a multi-tasking environment with the use of several levels of priority. Configuration of the processor and memory mapping for linking compiled code is as simple as specifying what memory segment to use for each 'section' of code. This configuration interface is used extensively.

In addition to configuration, DSP/BIOS allows debugging of the program running on the DSP by transferring register and memory contents to the host PC while the processor is temporarily halted.

The evolutionary program developed to run on this platform is called ga1 [Ferguson_GA1] and is broken into two main tasks, main and Supervisor Task. The main function, invoked by the task manager at DSP initialization, performs some initialization and then exits. The task manager then schedules Supervisor Task to run, which is the main routine for performing the GA. The EP routines described in Appendix C are called as part of Supervisor Task.

Programming:

The FPTA uses a synchronous programming interface with control signals CLEAR, RESET, ST, WR, DATA[15:0] and ADDR[8:0] and a clock input CLK. The interface is synchronized on CLK, which must be synchronized with the other control signals. During the programming of the FPTA, the stimulation is turned off, thus reducing overhead for the algorithm. The overall Program and Stimulate cycle can be seen in Figure 28 in which the Programming time is seen as a steady-state value of the Input signal $S(t_s)$.

The programming function for chromosome `i` follows this procedure:

```
/* Reset the configuration logic */
Raise the CLEAR line
Lower the RESET line
Cycle CLK
/* Output ADDR/DATA pairs for programming */
for(n=0; n<C; n++){
  for(word=0; word<5; word++) {
    DATA = pop[i].gene[n].geneData[word]
    ADDR = n×word
    Raise ST and WR
    Assert DATA and ADDR lines
    Cycle CLK
    Lower the ST and WR lines
    Cycle the CLK
  }
}
```

In order to minimize the time taken to download a chromosome, hand-coded assembly language has been written to access the memory-mapped I/O for the DIO, which eliminates several layers of system driver function calls.

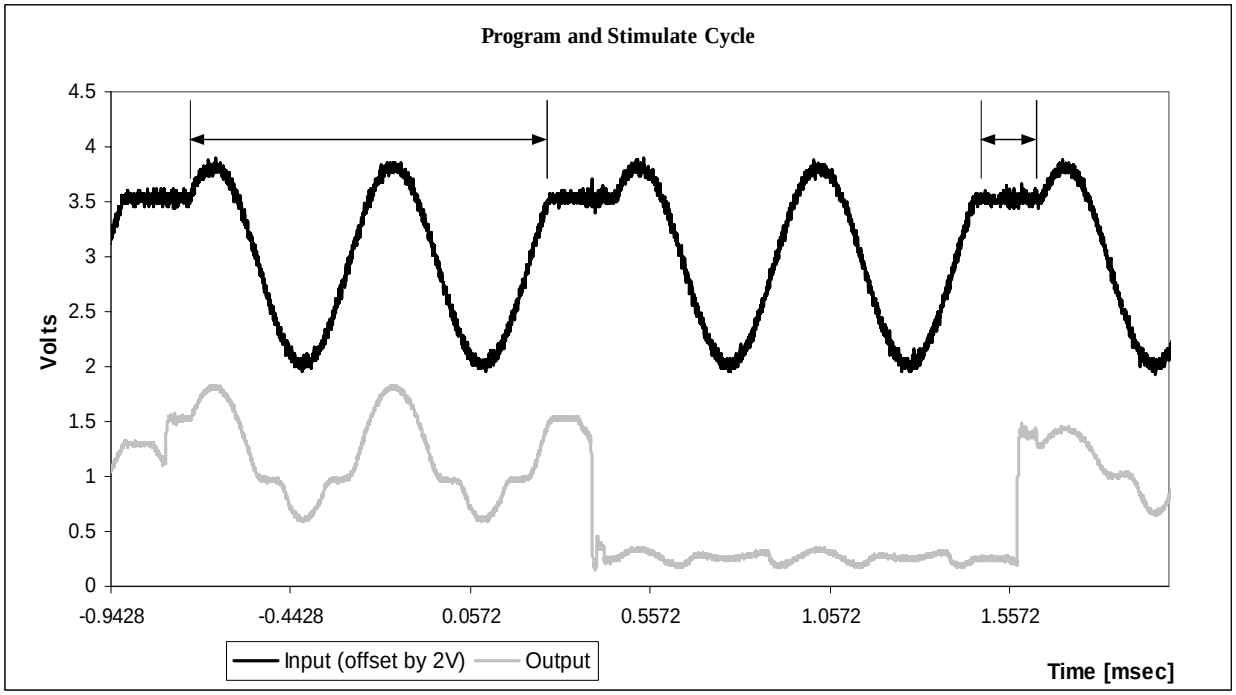


Figure 28 The Program and Stimulation cycle. The waveform exhibited was sampled at the maximum sampling rate (100kSamp/sec) corresponding to a minimum sample time t_s of 10 μs.

Stimul

Methodology of the stimulus/response cycle:

Each individual in the population is evaluated in a stimulus/response cycle performed as part of the GA. The cycle consists of a waveform of some specified function S(t) being input to the FPTA and the output of the chip R(t) being sampled n times and recorded as the response to be analyzed under the fitness criteria. This cycle is performed with the use of the SERVO-16 module in an interrupt driven mode. The method used to stimulate the circuit is derived from the architecture of the SERVO-16 and its FIFO memory structures. There are two FIFO memories, one for input and one for output, as seen in Figure 27, which are 32 bits wide and 256 entries deep. The 32 bits in each FIFO location are filled by two 16-bit samples, either input or output, an implication of this is that inputs and outputs are enabled in contiguous pairs (0:1, 2:3, etc.). The SERVO-16 control logic samples all enabled ADC pairs and updates all enabled DAC pairs at the same time, t, at some rate, r. The inputs/outputs are multiplexed to/from the respective FIFOs during the intervening sampling period. When an ADC pair is sampled, the lower numbered input is mapped to the low order 16-bits and the higher to the higher order 16 bits of the FIFO. If more than one pair is enabled the lower numbered pair is written to the FIFO first, followed by the next higher number, and so on. If the FIFO fills up then the next and subsequent samples are lost. When reading the ADC FIFO, if there are no entries in the FIFO, the last entry is read repeatedly. Similarly, if too many words are written to the DAC, the last samples are lost and if there are no entries in the FIFO, the last entry is output multiple times.

Interaction between the SERVO-16 and the DSP is initiated either when the DSP writes/reads data to/from the SERVO-16 or when the SERVO-16 issues an interrupt to the processor. This is because the DSP does not have access to the amount of data stored in the FIFOs and the only way to know the status of the FIFOs is to either reset the whole board or receive an interrupt from the SERVO-16. Interrupts are issued by the SERVO-16 based upon the number of entries in the one or both of the FIFOs based on one of three conditions, ADC_FIFO_HI (ADC is almost full, need to service - read), DAC_FIFO_LO (DAC is almost empty, need to service – write) or the logical OR of the two. The algorithm is currently written so that an interrupt occurs on DAC_FIFO_LO, which triggers when there are less than FIFO_THRESHOLD (currently 40) samples left in the DAC FIFO. When the interrupt occurs the interrupt service routine analog_isr is scheduled by the task manager and performs the FIFO updates. Since there are only 256 entries in the FIFO, a problem occurs when waveforms greater than 216 are employed. In this case there is a special section of code, which writes and reads segments of the waveforms to the DAC and from the ADC.

A time optimization is made by analyzing the $R(t_s)$ of one individual during the time the processor is waiting for the stimulus/response cycle for the next individual to complete. The time taken to complete a stimulus/response cycle is determined by the type of input, $S(t_s)$.

Appendix D Documentation for Supercomputer code

EHWPack Development System for Spice Software and FPTA Hardware User and Developer Manual, EHW Group, JPL. A.Stoica, G. Klimeck, D. Keymeulen, R. Zebulum and C. Lazaro.

This document presents the EHWPack development system, a tool that performs the evolutionary synthesis of electronic circuits, using the SPICE simulator and the Field Programmable Transistor Array hardware (FPTA) developed at JPL. We provide a user manual and a developer manual of the EHWPack tool. In the user manual, we first present a tutorial, in which a computational circuit is synthesized. Second we describe the Neptune supercomputer system, where this software tool is executed. In the developer manual, we describe the structure of the EHWPack software tool, both in terms of directories' organization and of functionality of the different modules. The last section of the developer manual describes the interface of the program with the SPICE simulator.

Appendix E Documentation for a series of evolved circuits

- Experiments with FPTA-2
    - Halfwave Rectifier (FPTA-2)
    - Low Pass Filters
    - Adaptive Band-Pass Filter
    - Automatic Gain Control (AGC) Circuit
    - Mixed Sine Wave Inputs
- Experiments with FPTA-1
    - Adder (FPTA-1)
    - Multiplier (FPTA-1)
- Experiments with FPTA-0
    - Recovery Functionality in Extreme Temperature (FPTA-0)
    - Fault-Recovery (FPTA-0)
    - Fuzzy Logics
- Unconstrained Evolution
    - Fuzzy Controller
    - Digital to Analog (DAC) Converter

Title of Experiment: **Halfwave Rectifier**

Experimental apparatus: Stand-alone EHW-DSP system

Reference: [IAN_GECCO02]

Objective:
The objective of this experiment is to evolve a halfwave rectifier given only two cells of the FPTA. The fitness function given below does a simple sum of error between the target function and the output from the FPTA. This is a very simple circuit to design by hand and was a proof-of-principle for the entire EHW/DSP platform.

Algorithm Parameters:

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{t_s=0}^{n-1} \begin{cases} R(t_s) - S(t_s) & \text{for } (t_s < n/2) \\ R(t_s) - V_{\max}/2 & \text{otherwise} \end{cases}$
F_t	Target fitness	4500
$S(t_s)$	Stimulus waveform	Single Sine wave (2kHz, V_{max}=1.8V)
n	Number of samples	50
N	Population size	100
C	Number of genes (FPTA cells) used	2
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

Solution Waveform:

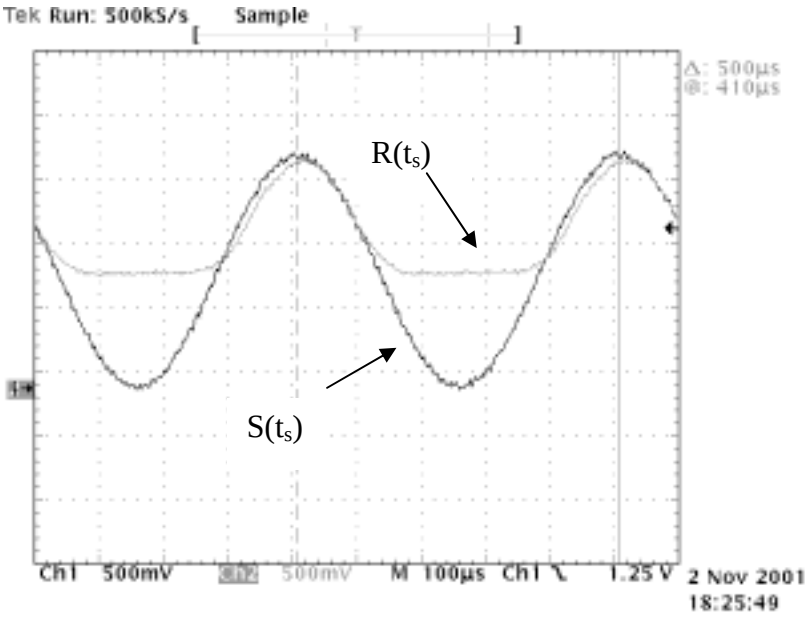


Figure 29 The solution for the Halfwave rectifier.

Analysis

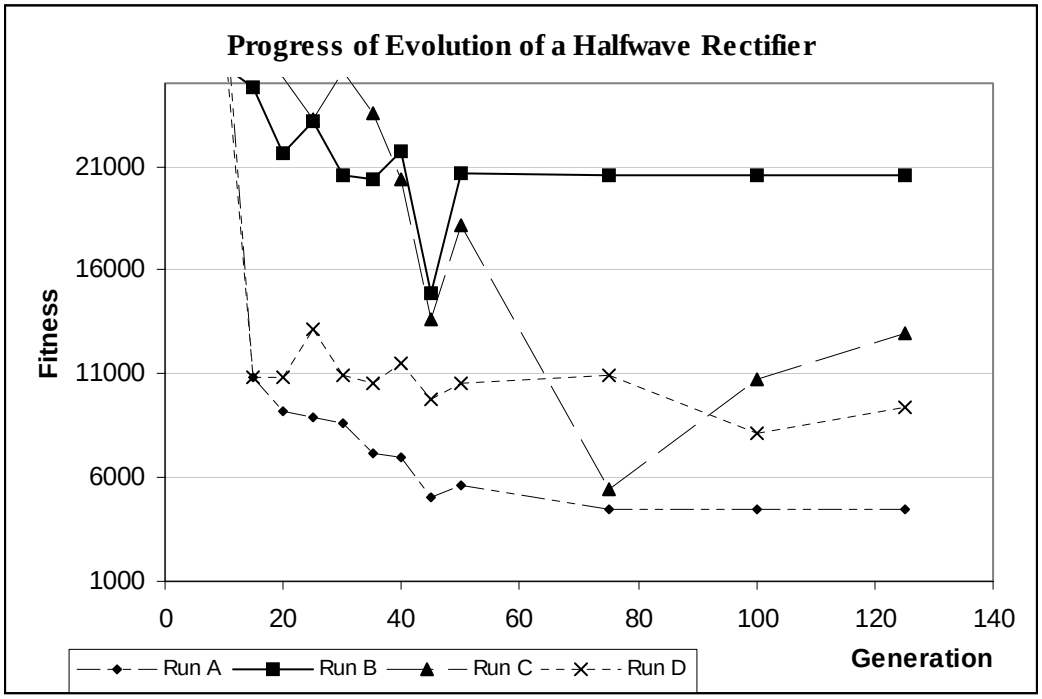


Figure 30 The fitness function as generations progress. The first few generations showed fitness values near 100,000 and are not shown on this scale.

Observations:

It is observed that individual solutions programmed on the FPTA suffer somewhat from an apparent instability, which arises when the evaluation of a given individual depends on the previous state of the FPTA. Often the best individual from a given population does not perform as well when stimulated independently from the others. Nevertheless, evolution weeds these individuals out and solutions are almost always found within the first 50 generations, or about 20 seconds.

The results shown in the graph above are typical of a series of successful runs. Approximately 1 out of 10 runs ended with the algorithm getting stuck and not finding a solution at all using the fixed mutation rate of version 1.0 of the software.

Title of Experiment: **Low Pass Filter**

Objective:

The objective of this experiment is to evolve a low-pass given four cells of the FPTA2. The fitness function given below performs a sum of error between the target function and the output from the FPTA in the frequency domain. Given two tones, the objective is to have at the output only the lowest frequency tone (10kHz). This hardware evolved circuit demonstrated that the FPTA2 is able to realize active filters with some gain.

Algorithm Parameters: (R(f) is the circuit response, T(f) is the target)

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{t_s=0}^{n-1} (R(f) - T(f))$
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	Two sine waves (10kHz, 25kHz)
n	Number of samples	50
N	Population size	400
C	Number of genes (FPTA cells) used	4
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

Solution Waveform:

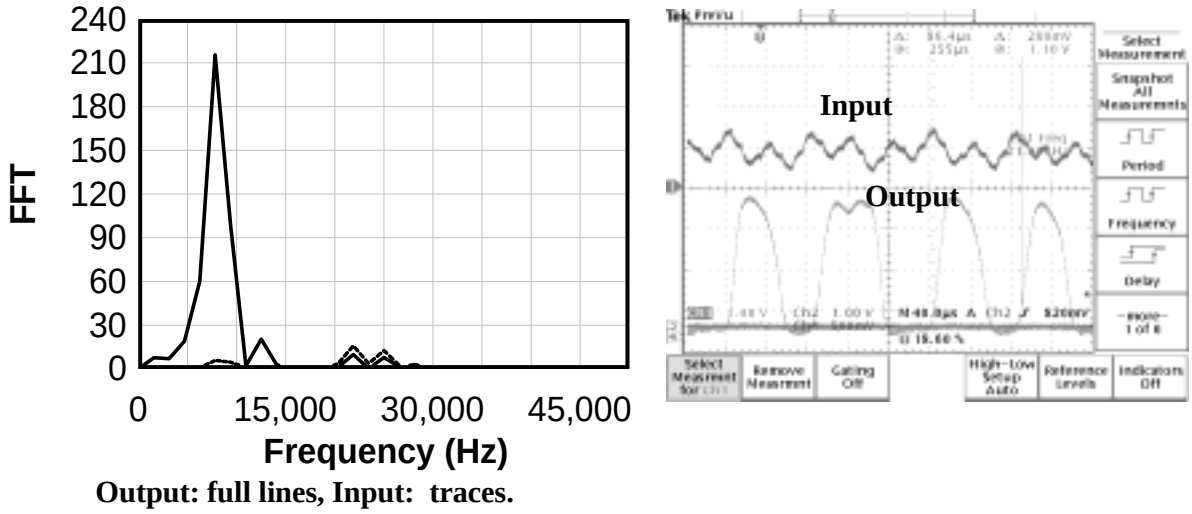


Figure 33: Low-Pass Filter. The graph in the left displays the FFT of the input and output signals. The graph in the right shows the circuit input stimulus and response in the time domain (Circuit stimulated by two sine waves: 10kHz and 25kHz).

Observations:

Observations: It was measured a gain of 15.6dB for the desired frequency (10kHz), falling to 21.dB or the undesired tone (25kHz).

Title of Experiment: **Adaptive Band-Pass Filter**

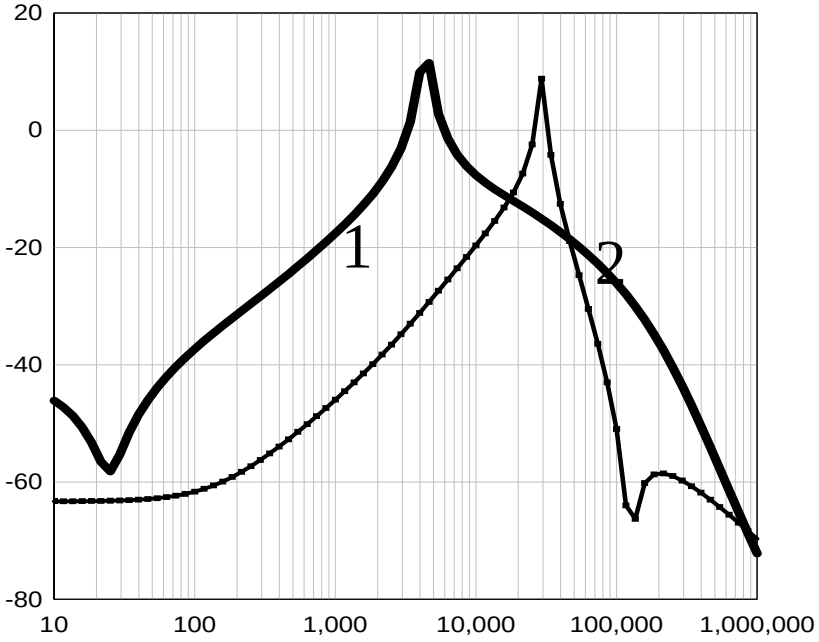
Objective:

The objective of this experiment is to evolve adaptive band-pass filters using SPICE simulation models of the FPTA. The evolved circuit is adaptive in the sense that *the same re-configurable circuitry* can be evolved to realize different filter specifications. Particularly, band-pass filters with center frequencies at 5kHz and 20kHz respectively have been evolved. The fitness function comprises a weighted sum of errors between the target frequency response and the circuit output in the frequency domain. The weights are higher for frequency points inside the passing band.

Algorithm Parameters: : ($R(f_c)$ is the circuit response, $T(f_c)$ is the target, in the frequency domain)

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{f_s=0}^{n-1} w_f (R(fc) - T(fc))$
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	Small signal simulation of SPICE
n	Number of samples	200
N	Population size	128
C	Number of genes (FPTA cells) used	3
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.5

Solution Waveform:



Filter 1 (5kHz)

Gain = 11.4dB

Roll-off: 34dB/dec, -30dB/dec

Filter 2 (20kHz)

Gain = 9dB

Roll-off: 43dB/dec, -70dB/dec

Figure 31 The solution for the low-pass filter.

Analysis

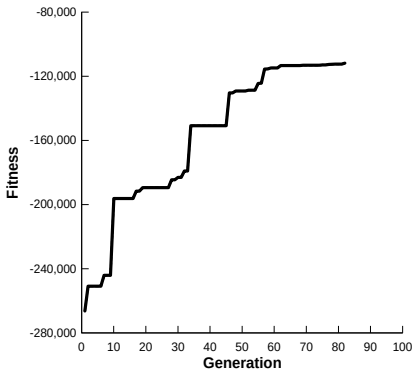


Figure 32 The fitness function as generations progress .

Observations:

The experiment was executed on a 128 nodes parallel machine. This particular experiment encompassed 16 processors, each one simulating the SPICE netlist corresponding to one individual. The experiment lasted around 10 minutes, processing about10^4 circuit netlists. The fitness functions utilized large weight values to increase the penalty on the errors; therefore, although the final fitness was still a large negative value, the evolved circuits achieved the desired specifications (Gain around 10dB in the passing band, roll-off about 40dB/dec). The FPTA model is able to adapt the filter response to different fitness specifications without using programmable capacitors, as in conventional Field Programmable Analog Arrays (FPAAs). In this experiments, a total of 12 fixed-value capacitors were used (3 cells with 4 capacitors each).

Title of Experiment: **Automatic Gain Control (AGC) Circuit**

Objective:

The objective of this experiment is to evolve a circuit that can provide some degree of gain control, e.g., large gain for low amplitude inputs and small gain for high amplitude inputs. AGCs are a very important building block in communication circuits and it has been verified in this experiment that four FPTA cells can accomplish this task. Two sine waves of different amplitudes were applied as stimulus and the target output was a sine wave of constant amplitude. The fitness encompassed the absolute sum of errors between the FPTA output and the target.

Algorithm Parameters: (R(t) is the circuit response, $T(t_c)$ is the target, in the time domain)

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{t=0}^{n-1} \mid (R(t) - T(t)) \mid$
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	Two sine waves (0.15V and 0.3V respectively)
n	Number of samples	50
N	Population size	700
C	Number of genes (FPTA cells) used	7
P_c	Crossover probability	0.2
P_m	Mutation probability	0.1
P_e	Elite percentage to save	0.2

Solution Waveform:

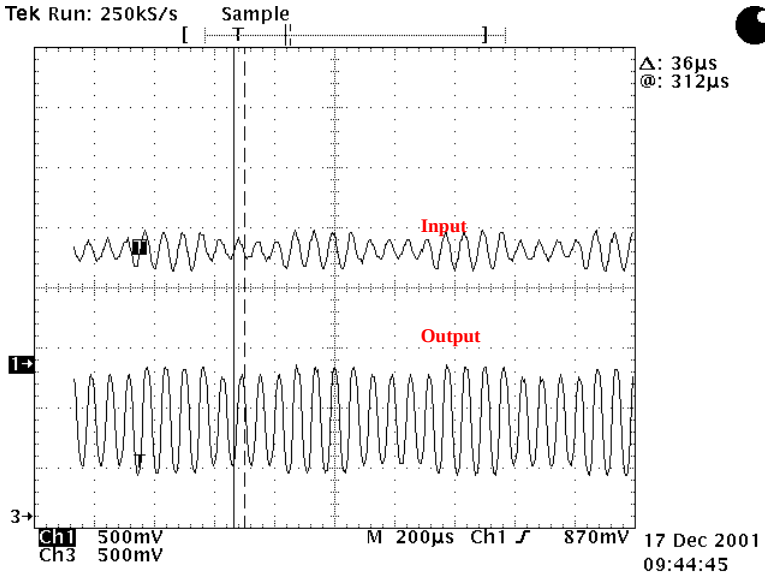


Figure 33 The solution for the AGC circuit

Observations:

The experiment was performed directly in hardware, taking around 5 minutes. The circuit provided a gain of 5 for the smaller sine wave, decreasing to 2.8 for the larger sine wave. The initial ratio 2:1 of the input signals has been reduced to 1.13 :1 at the circuit output.

Title of Experiment: **Mixed Sine Wave Inputs**

Objective:

The objective of this experiment is to show that the FPTA cells can evolve into a circuit that can automatically recognize and filter the strongest signal between two sine waves. These two tones (4kHz and 10kHz) were linearly combined by a mixing matrix to produce the circuit inputs, $E_1(t)$ and $E_2(t)$.

$$\begin{bmatrix} E_1(t) \\ E_2(t) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 \\ 0.25 & 0.5 \end{bmatrix} \begin{bmatrix} \sin(2\pi 4000.t) \\ \sin(2\pi 10000.t) \end{bmatrix}$$

The experiment comprises two evolutionary runs: the first in which $E_1(t)$ is applied at the FPTA2 input and the target is the 4kHz tone; a second run in which $E_2(t)$ is applied at the FPTA2 input and the target is the 10kHz tone. The fitness function is the same in both experiments, i.e., the Genetic Algorithm "does not know" which is the input combination. The fitness is computed by taking the FFT of the circuit output. The highest component of the FFT of the circuit output is found and the objective is to maximize it, or minimize the expression defined in the table below.

Algorithm Parameters (R(f) is the FFT component of the circuit response R at the frequency f):

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = 50 - Max_f[R(f)]$
F_t	Target fitness	30
$S(t_s)$	Stimulus waveform	Combined sine wave consisting of 4kHz and 10kHz tones)
n	Number of samples	50
N	Population size	500
C	Number of genes (FPTA cells) used	4
P_c	Crossover probability	0.7
P_m	Mutation probability	0.2
P_e	Elite percentage to save	0.1

Solution Waveform:

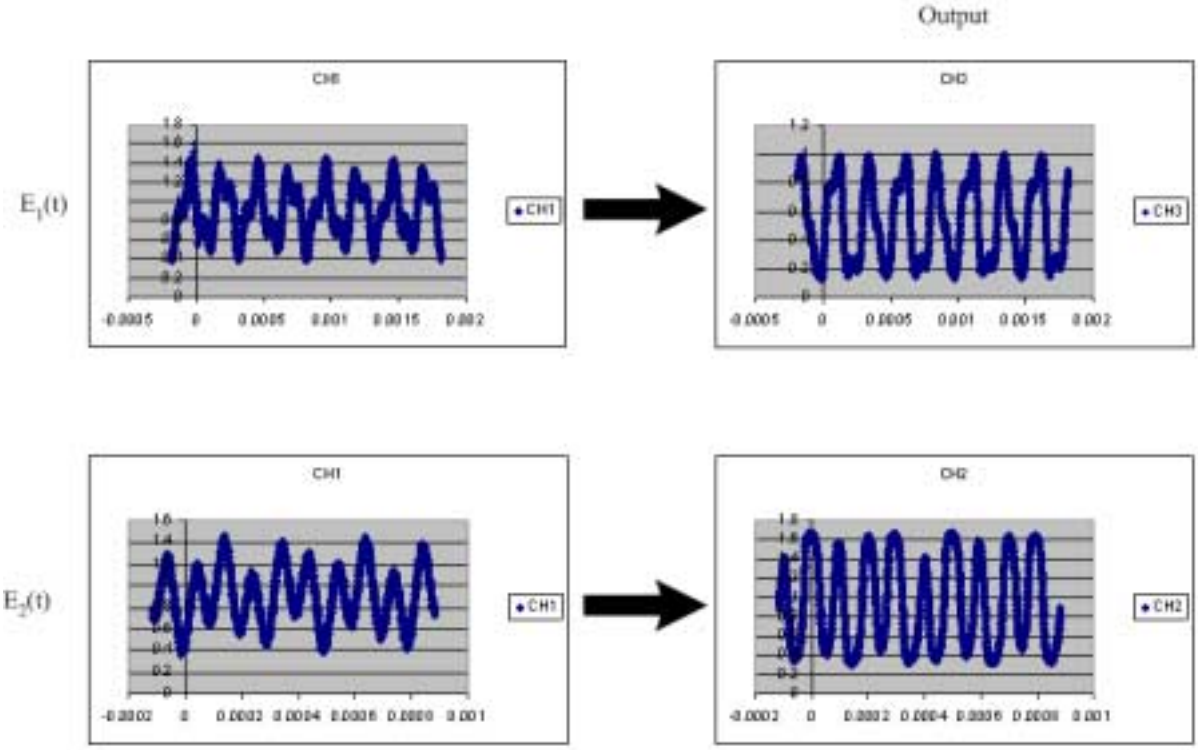


Figure 34 The solution for the Mixed Sine Wave problem.

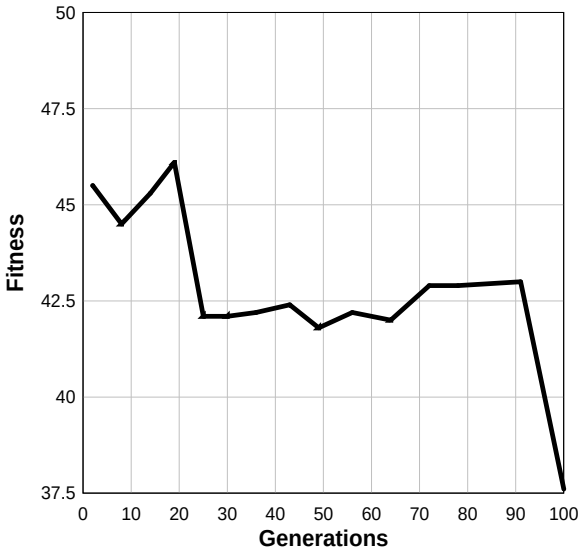


Figure 35 The fitness function as generations progress .

Observations:

When $E_2(t)$ was applied as an input to the circuit, the circuit would amplify both tones. However, the unwanted tone (4kHz) was amplified by 2.7dB, while the desired tone was amplified by 4dB. When $E_1(t)$ was applied to the circuit, the unwanted tone (10kHz) was not attenuated nor amplified, but the desired tone (4kHz) was amplified about 3dB. These are still preliminary and recent results, and there is a lot of room for improvement. Nevertheless, they are the first hardware evolved adaptive filters, in which the circuit can *automatically* recognize the desired tone and (partially) eliminate the unwanted tone.

Title of Experiment: **Adder**

Objective:

The objective of this experiment is to evolve an analog adder using the FPTA1 chip. The fitness function given below does a simple sum of the square of the errors between the target function and the output from the FPTA1.

Algorithm Parameters :

$T(t_i)$ is the target waveform determined ahead of time.

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{i=0}^{W} (R(t_i) - T(t_i))^2$ where W is window size
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	2 sine wave generated by LabView -inp1: 1.4kHz, Amp 1.0V ($2.0V_{p-p}$), offset 1.6V -inp2: 7.6kHz, Amp 1.1V ($2.0V_{p-p}$), offset 1.6V
n	Number of samples	64
N	Population size	200
C	Number of genes (FPTA cells) used	6
P_c	Crossover probability	0.8
P_m	Mutation probability	0.1
P_e	Elite percentage to save	0.05

Solution Waveform:

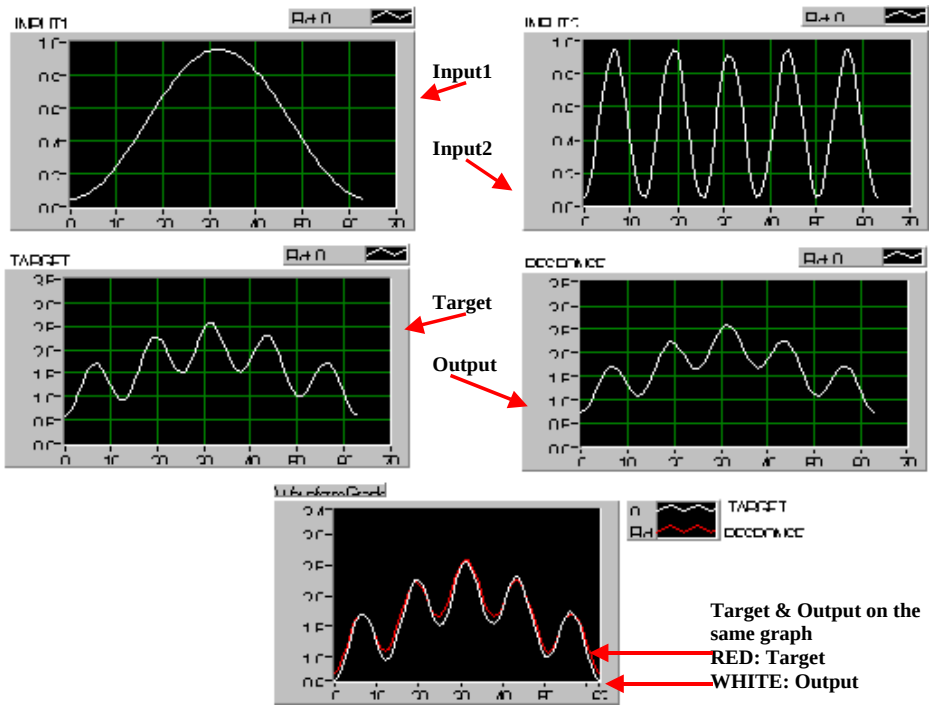


Figure 36 - Waveforms of the Adder experiment: Inputs, output and target

Analysis

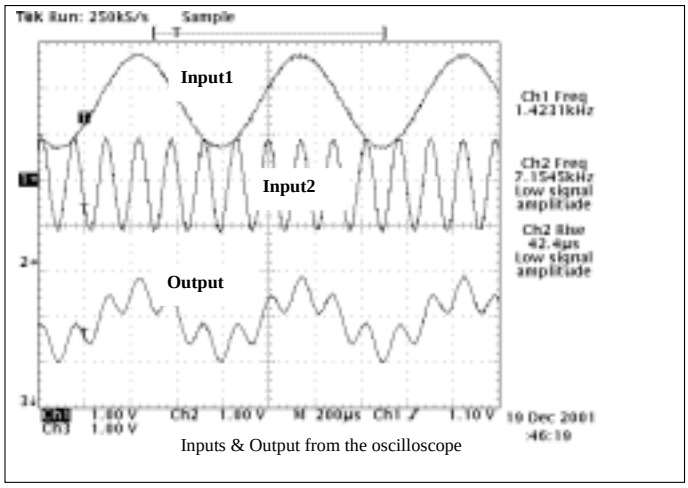


Figure 37 – Detailed view of two inputs and output of the adder circuit.

Observations

- Frequency/amplitude dependent?
- Number cells used
- Multiple inputs and outputs selected by the GA?

Title of Experiment: **Multiplier**

Objective:

The objective of this experiment is to evolve an analog multiplier using the FPTA1 chip. The fitness function given below does a simple sum of the square of the errors between the target function and the output from the FPTA1.

Algorithm Parameters

$T(t_i)$ is the target waveform determined ahead of time.

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{i=0}^{W} (R(t_i) - T(t_i))^2$ where W is window size
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	2 sine wave generated by LabView -inp1: 1.4kHz, Amp 1.1V ($2.0V_{p-p}$), offset 1.6V -inp2: 7.6kHz, Amp 1.1V ($2.0V_{p-p}$), offset 1.6V
n	Number of samples	64
N	Population size	200
C	Number of genes (FPTA cells) used	6
P_c	Crossover probability	0.8
P_m	Mutation probability	0.1
P_e	Elite percentage to save	0.05

Solution Waveform:

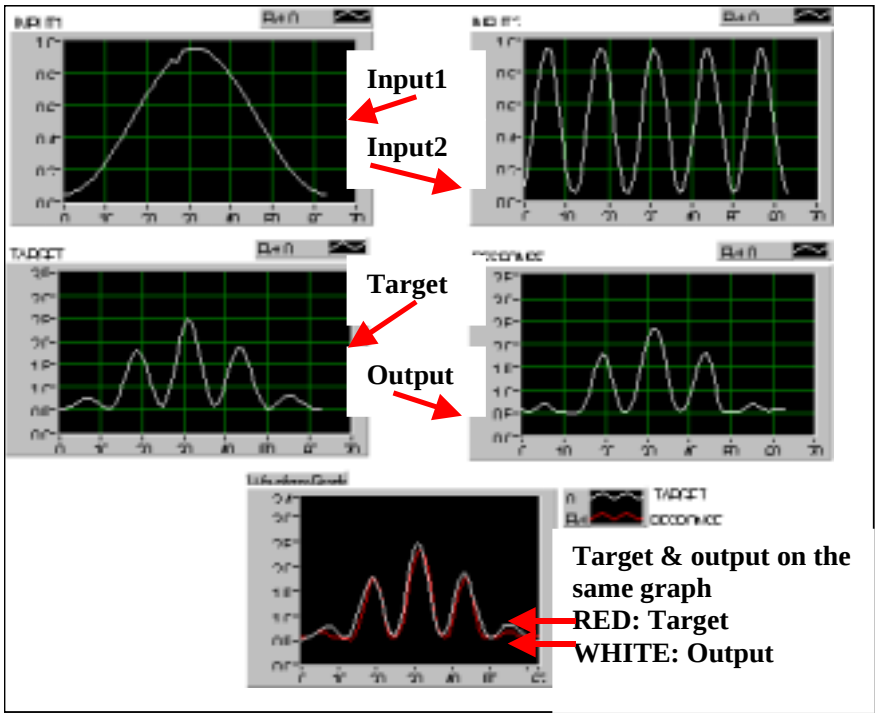


Figure 38 - Inputs, output and target waveform for the multiplier.

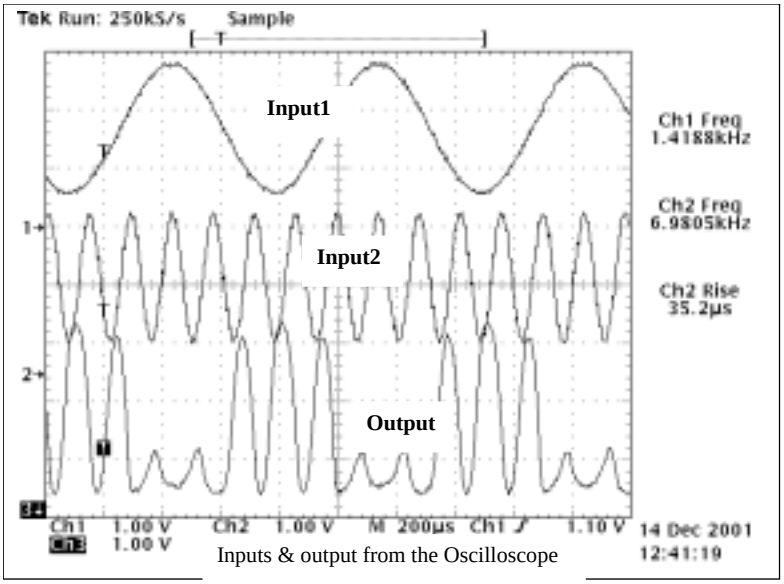


Figure 39 - Detailed view of multiplier input/output waveforms.

Observations

The best chromosome obtained after 200 generations is with the associated mean square error 1.3e-2 and the evolving time 4701 seconds.

Title of Experiment: **Recovering Functionality at Extreme Temperature**

Reference: [Stoica_TEMP01] [Stoica_NASA01]

Objective:

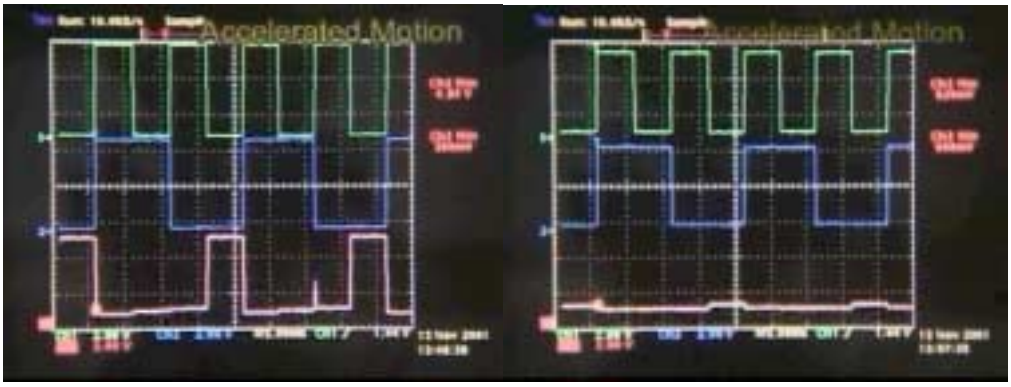
The objective of this experiment is to recover functionality of the FPTA-0 under extreme temperature of –197 °C and 320 °C. The fitness function given below does a simple sum of errors between the target function and the output from the FPTA-0. The functionality of the FPTA-0 are AND, NAND, NOR.

Algorithm Parameters:

$T(Sn_i, Sn_j)$ is the NOR target waveform determined ahead of time.

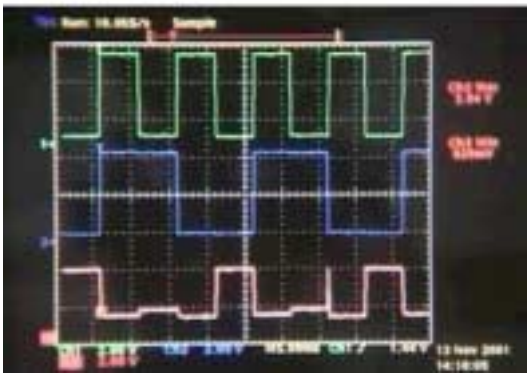
Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{Sn1} \sum_{Sn2} \lvert R(Sn1, Sn2) - T(Sn1, Sn2) \rvert$
F_t	Target fitness	4500
$S(t_s)$	Stimulus waveform	Step wave (50 Hz, V_{max}= 5V) Step wave (100 Hz, V_{max}= 5V)
n	Number of samples	10
N	Population size	100
C	Number of genes (FPTA cells) used	1
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

Degraded Functionality:

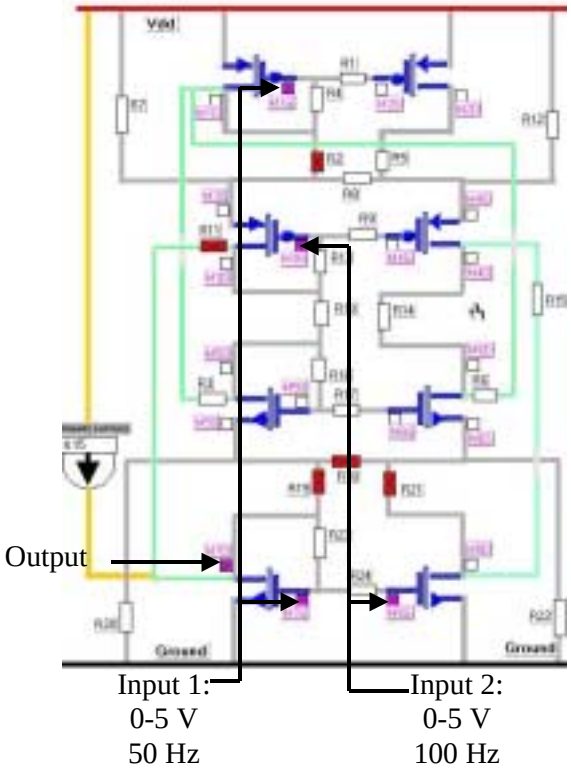


Original NOR gate at 27 C Degrade NOR gate at 326 C

Solution Waveform:



NOR gate recovery at 329C



Analysis

The results shown in the graph above are typical of a series of successful runs.

Observations:

1. Mutant solutions exist at extreme temperature.
2. Solutions obtained by evolution at extreme temperature degrade at room temperature.

Title of Experiment: **Fault recovery**

Reference: [Keymeulen_IEEEJour00]

Objective:

These experiment show results obtained using population based approaches for the synthesis of a fault-tolerant digital circuit (XNOR) on FPTA-0 and a fault-tolerant analog circuit (multiplier) using SPICE simulator of FPTA-0. Our experiments show that the evolutionary algorithm is capable of synthesizing fault-tolerant designs for both the analog and digital functions circuits that can recover for functionality when lost due to not a-priori known faults by finding new circuits configurations that circumvent the faults.

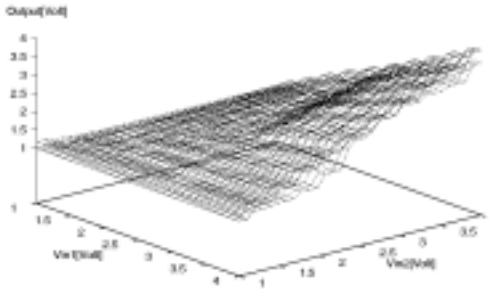
Algorithm Parameters:

$T(t_i)$ is the multiplier target waveform determined ahead of time.

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{i=0}^{W} (R(t_i) - T(t_i))^2$ where W is window size
F_t	Target fitness	4500
$S(t_s)$	Stimulus waveform	2 sine wave generated by LabView - S_1: 1.4kHz, Amp 4.0V - S_2: 7.6kHz, Amp 4.0V
n	Number of samples	10
N	Population size	100
C	Number of genes (FPTA cells) used	1
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

Degraded Functionality for multiplier:

Best individual at generation 59



Fault injected at generation 60

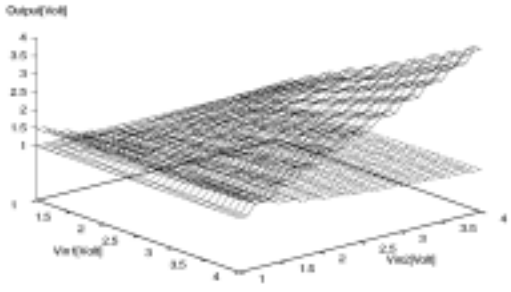


Figure 40 – Best individual before and after fault injection.

Solution Waveform for multiplier:

Self-repaired individual at generation 80

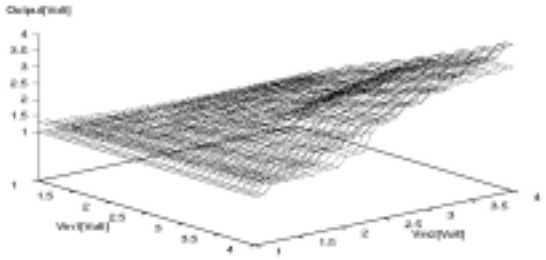


Figure 41 - Recovered individual for the multiplier.

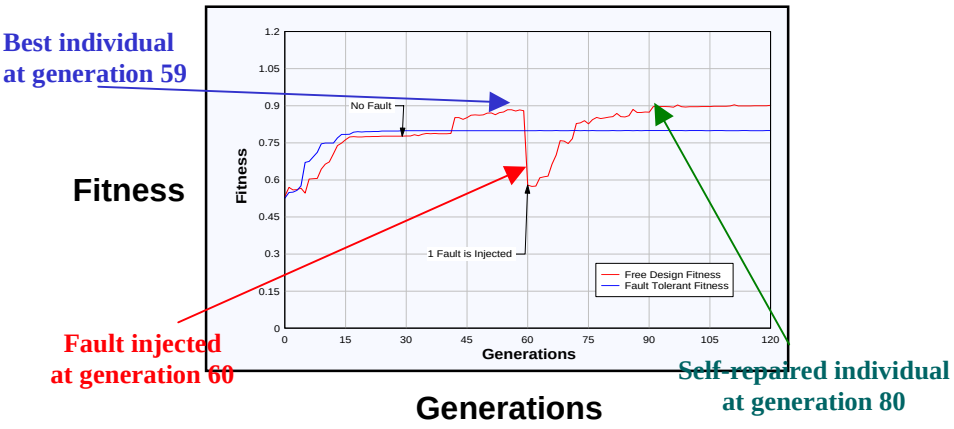


Figure 42 - Fitness along generations.

Observations:

- Classical fault-tolerant electronic devices rely on redundancy at system/subsystem level and knowledge of potential faults characteristics while the evolvable hardware technology automatically recovers functionality, copes with unidentified faults by restarting the evolution process.

Title of Experiment: **Fuzzy Logics**

Objective:

The objective of this experiment is to evolve circuits implementing parametrical connectives for fuzzy logics [Stoica_EH00]. These circuits are unconventional ones, for which there are no textbook design guidelines, are particularly appealing to evolvable hardware. Frank's parametric T-norms and T-co norms (also referred to as fundamental T-norms/co norms) are the selected choice for modeling the logical connectives. Particularly, the family of Frank T-norms is given by

$$T_s(x,y) = \begin{cases} MIN(x,y) & \text{if } (s = 0) \\ x \cdot y & \text{if } (s = 1) \\ \log_s\left[1 + \frac{(s^x - 1).(s^y - 1)}{s - 1}\right] & \text{if } (0 < s < \infty), s \neq 1 \\ MAX(0, x + y - 1) & \text{if } (s = \infty) \end{cases}$$

Algorithm Parameters: (R(In2, In2) is the circuit response, T(In1, In2) is the target, In1 and In2 are the circuit inputs)

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{In1} \sum_{In2} \left\| R(In1, In2) - T(In1, In2) \right\|$
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	Double DC sweep of In1, In2 from 1 to 4V
n	Number of samples	100
N	Population size	128
C	Number of genes (FPTA cells) used	2----------
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	-------

Solution Waveform:

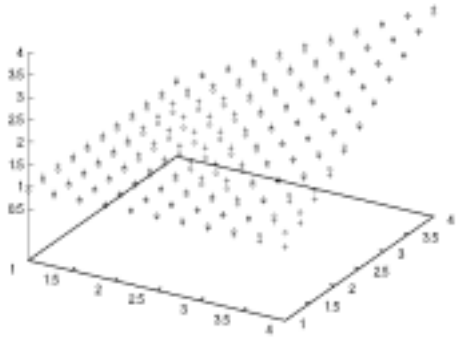


Figure 43 Response of a circuit implementing the fundamental T-norm for s=100 (◊). Target characteristic shown with (+).

Analysis

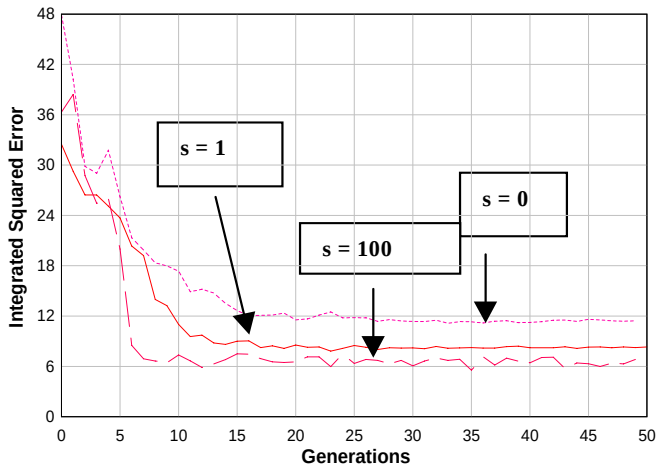


Figure 44 - Decreasing error between best individual in each generation and target circuit, for the three software evolved circuits, with s=0, s=1, s=100.

Observations:

The Mean Absolute Percentage Error (MAPE) to the target response stayed at 3.6% for the experiment shown above (T-norm s = 100). Experiments were performed for other values of s, and the highest error found was of 9%.

Title of Experiment: **Fuzzy Controller**

Objective:

The objective of this experiment is to evolve an approximation to a Fuzzy Controller using SPICE simulation models. The fitness function given below computes a sum of errors between the target function obtained as points on the control surface derived by a set of fuzzy control rules and the output from the circuit. We were able to achieve a circuit that approximates the control surface of a 2-input fuzzy controller, mapping thus a full fuzzy system in only seven transistors [Zebulum_IEEEAero02].

Algorithm Parameters: (R(In2, In2) is the circuit response, T(In1, In2) is the target, In1 and In2 are the control inputs)

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \sum_{In1} \sum_{In2} \left\| R(In1, In2) - T(In1, In2) \right\|$
F_t	Target fitness	0
$S(t_s)$	Stimulus waveform	Double DC sweep of In1, In2 from 0 to 3.3V
n	Number of samples	225
N	Population size	50
C	Number of genes (FPTA cells) used	----------
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

Solution Waveform:

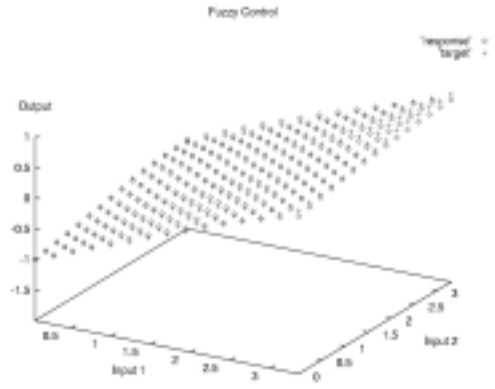


Figure 45 The solution for the Fuzzy controller.

Analysis

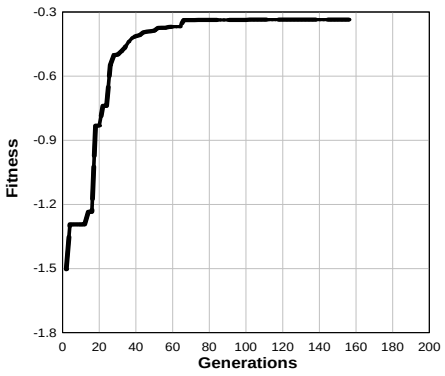


Figure 46 The fitness function as generations progress.

Observations:

- A compact circuit, using only seven transistors and operating in current mode, was evolved. The average error achieved was of 1.93%, and the maximum error to the target surface was 6.7%. The circuit is rather robust, and was tested at variations in transistor sizes, supply voltage and temperature, with the following results: decreasing the transistor sizes by a factor of 10 did not change the circuit response and the deviation from the target; average error of 1.98% and maximum error of 6.96% when decreasing the power supply voltage to 4.75V; average error of 1.94% and maximum error of 6.65% when increasing the power supply voltage to 5.25V; average error of 1.89% and maximum error of 6.3% when decreasing the temperature to 0^{o}C; average error of 1.98% and maximum error of 7.2% when increasing the temperature to 55^{o}C.

Title of Experiment: **Hierarchical Evolution of Digital to Analog Converters (DAC)**

Objective:

The objective of this experiment was to synthesize a 4-bit DAC *hierarchically*. At first, a 2-bit DAC was evolved from scratch, e.g., using MOS transistors as components for evolution . This circuit was then employed as a building block for evolution to synthesize a 3-bit DAC, which was subsequently used as building block for evolution to synthesize a 4-bit DAC [Zebulum_IEEEAero01].

Another distinct feature of this experiment is that the fitness function did not consist of the traditional sum of errors of the response to the target. Instead, the fitness evaluation function described in the table below rewarded circuits presenting uniform stepwise output. For each state transition i of the DAC, we measure the gradient in the current output ΔI_i. The average $<\Delta I_i>$ and Mean Squared Error (MSE) to the average value, $MSE(\Delta I_i , < \Delta I_i >)$, are computed over the 15 state transitions.

Algorithm Parameters:

Parameter	Description	Value
$F(t_s)$	Fitness Function	$F = \langle \Delta I_i \rangle - k \sum_i (\Delta I_i - \langle \Delta I_i \rangle)^2$
F_t	Target fitness	----------
$S(t_s)$	Stimulus waveform	4 digital inputs comprising 16 different configurations
n	Number of samples	100
N	Population size	50
C	Number of genes (FPTA cells) used	----------
P_c	Crossover probability	0.7
P_m	Mutation probability	0.04
P_e	Elite percentage to save	0.2

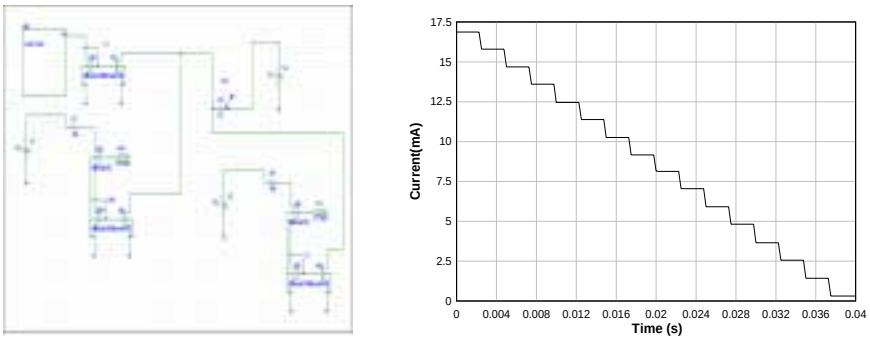


Figure 47 Schematic and circuit response for the 4-bit DAC.

Analysis

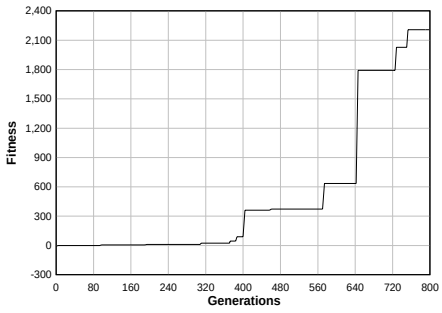


Figure 48 The fitness function as generations progress. High fitness values indicate better circuits.

Observations:

As the target circuit specification gets more complex, the circuit synthesis task through evolution gets more difficult as well. This is known as the scalability problem, e.g., the performance deterioration of Evolutionary Systems when synthesizing more complex circuits. This experiment successfully tests a methodology to cope with this problem, by letting evolution use high level building blocks or re-use previously evolved building blocks to synthesize more complex circuits.

The evolved 4-bit DAC presented also a good performance in terms of differential non-linearity (DNL), the maximum being 11% higher than the LSB and also a maximum settling time of 10ns.

Appendix F Other EHW Testbeds

1. Extreme Temperature Environment Test Bed

The purpose of this testbed is to achieve temperatures exceeding 250 °C on the die of the FPTA-0 while staying below 250 °C on the package. It is necessary to stay below 250 °C on the package in order not to destroy the interconnects and package integrity. Die temperatures should stay below 400 °C to make sure die attach epoxy does not soften and that the crystal structure of the aluminum core does not soften. To achieve these high temperatures the testbed includes an Air torch system and a ThermaCAM camera. The Air Torch is firing hot compressed air through a small hole of a high temperature resistance ceramic protecting the chip. To measure temperature an infrared camera (ThermaCAM) and Thermocouples were used.

Figure 49 shows the Air Torch apparatus electronically controlled by PID controller, which maintains a precision of ±10° C up to 1000° C. Figure 50 shows also the ceramic protecting the die connections and the package. The Temperature was measured above the die and under the die using thermocouples.

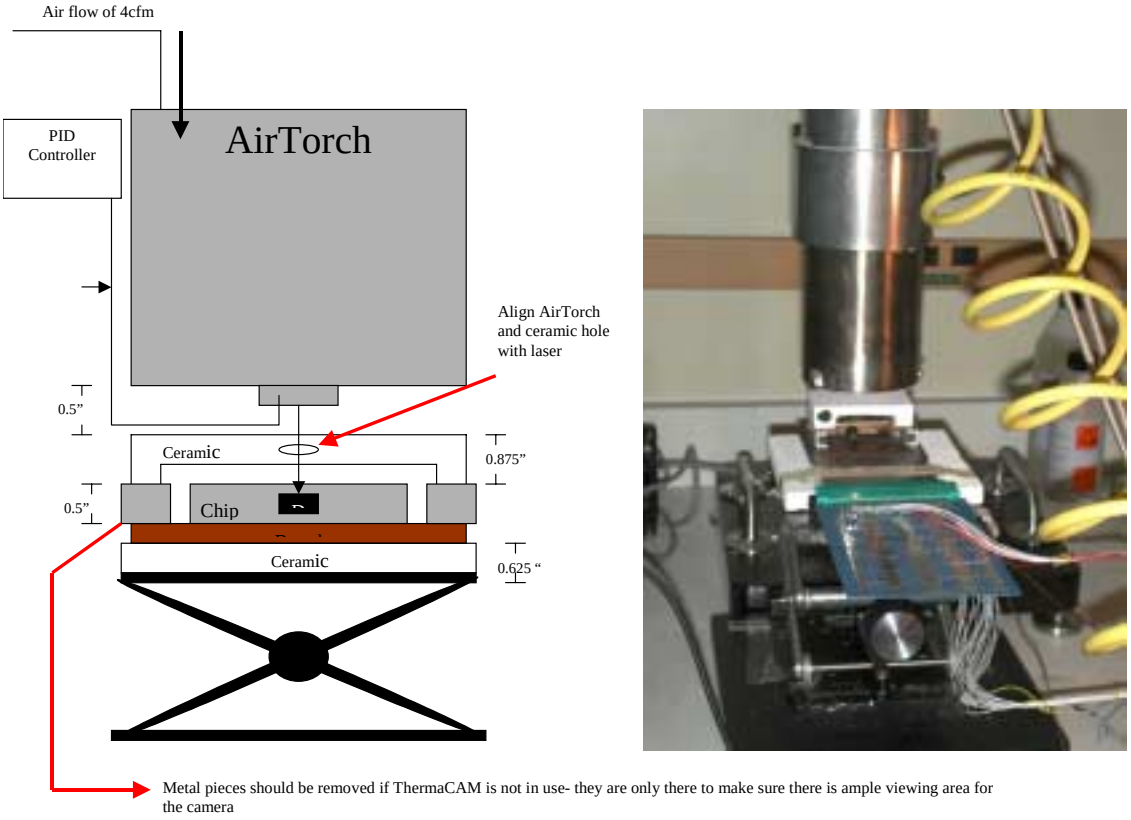


Figure 49 Experimental Setup for Extreme Temperature Experiments

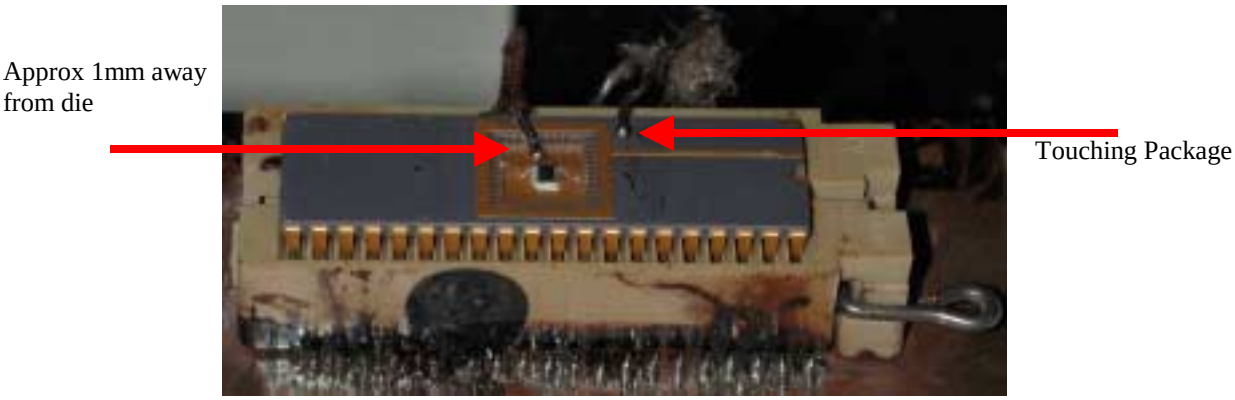


Figure 50 Measurement Techniques using thermocouples above and under the die

Reference: [Stoica_TEMP01][Stoica_MAPLD00]

2. Mixtrinsic – architecture/communication w/supercomputer and hardware

The EHWPack was implemented on the HP Exemplar shared-memory supercomputer with 256 CPUs and 64 GB of memory of the Jet Propulsion Laboratory. The exemplar is composed of 16 nodes, with each node having 16 processors. One node, the single-node System sub complex, is dedicated to the users login sessions and compilation jobs. All the other nodes are reserved for batch jobs or interactive jobs requiring input-output interface with the user during execution.

The parallel programming model used by EHWPack is the one implemented by the PGAPack. It is a message passing model, in particular the Single Program Multiple Data (SPMD) model using the Message Passing Interface (MPI) standard implemented by the Neptune software libraries. The EHWPack parallel implementation uses a master/slave algorithm, in which one process, the master, executes all steps of the genetic algorithm except the function evaluations (SPICE simulation or evolvable hardware evaluation). Slave processes execute the function evaluations. The master is running on one processor of the HP exemplar and the slaves are running on the other processors allocated for the job or on the FPTA hardware through an internet communication.

The hardware configuration is a board mounted with four FPTAs. The board is controlled by National Instruments data acquisition hardware and software (LabView). The LabView Software implements a tcp-ip client-server system where the server is the LabView and the master processor running on neptune is the client. LabView receives the configuration bits from the master and returns the sampling data of the output responses back to the master.

This environment allows therefore the implementation of the Mixtrinsic Evolution concept, since some processors are going to run SPICE simulations using the FPTA simulation model, while others will download the circuit configuration directly on the FPTA hardware and acquire the actual circuit response.

The graphical display is executed on a local UNIX workstation through a Xwindows interface.

Reference: [Stoica_ICES01]

3. DAQ boards and LabView GUI

An evolutionary design tool was developed to facilitate experiments in hardware evolution and is illustrated in Figure 51. The tool uses the EHWPack software package and also an evolvable hardware test bed built around LabView. The EHWPack was developed by our group and includes the public domain Parallel Genetic Algorithm package, PGAPack. An interface code based on TCP/IP protocol links the GA with the hardware where potential designs are evaluated, while a GUI allows easy problem formulation and visualization of results. At each generation the GA produces a new population of binary chromosomes, which get converted into configuration bits for the reconfigurable devices. LabView downloads configuration bits into the hardware device.

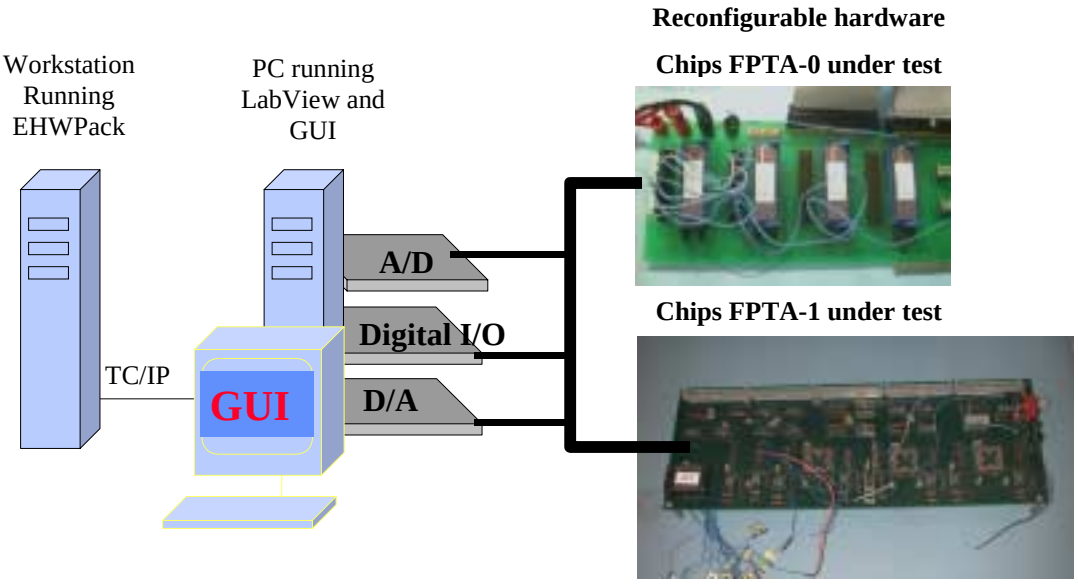


Figure 51 Hardware Evaluation Platform for FPTA-0 and FPTA-1.

The hardware testbed built around LabView includes 3 data acquisition boards connected to FPTA-0 and FPTA-1 and controlled by LabView drivers:

- National Instrument AT-AO-10 board with ten analog output channels, double-buffered, multiplying, 12 bit DACs; unipolar and bipolar voltage output, 4 to 20 mA current output, an on board DAC reference voltage of 10V, internal timer and external signal update capability for waveform generation, an on board 1,024-word FIFO buffer, transfer rate up to 200ksamples/sec per channel, on board analog output auto calibration circuitry, eight digital I/O lines able to sink up to 24mA of current, timer-generated and externally generated interrupts, a high performance RTSI bus interface, and full PC I/O channel DMA capability with analog output.

- National Instrument PCI-DIO-96 board, which is a 96-bit, parallel, digital I/O, interface for PCI bus computers. Four 82C55A programmable peripheral interface (PPI) chips control the 96 bits of TTL-compatible digital I/O. The four 82C55A PPI chips can operate in unidirectional mode, bi-directional mode, or handshaking mode and can generate interrupt request to a host computer.

- National Instrument AT-MIO-64E-3 board which is a completely Plug and Play-compatible multifunction analog, digital, and timing I/O boards for the PC AT and compatible computers, with 16 bit ADCs, 64 analog inputs, 16 bit DACs, 32 lines of TTL-compatible digital I/O, and 24-bit counter/timers for timing I/O.

Reference: [Keymeulen_GECCO00]